State-of-the-Art Data Mining Using Random Forests

Leo Breiman

U.C. Berkeley

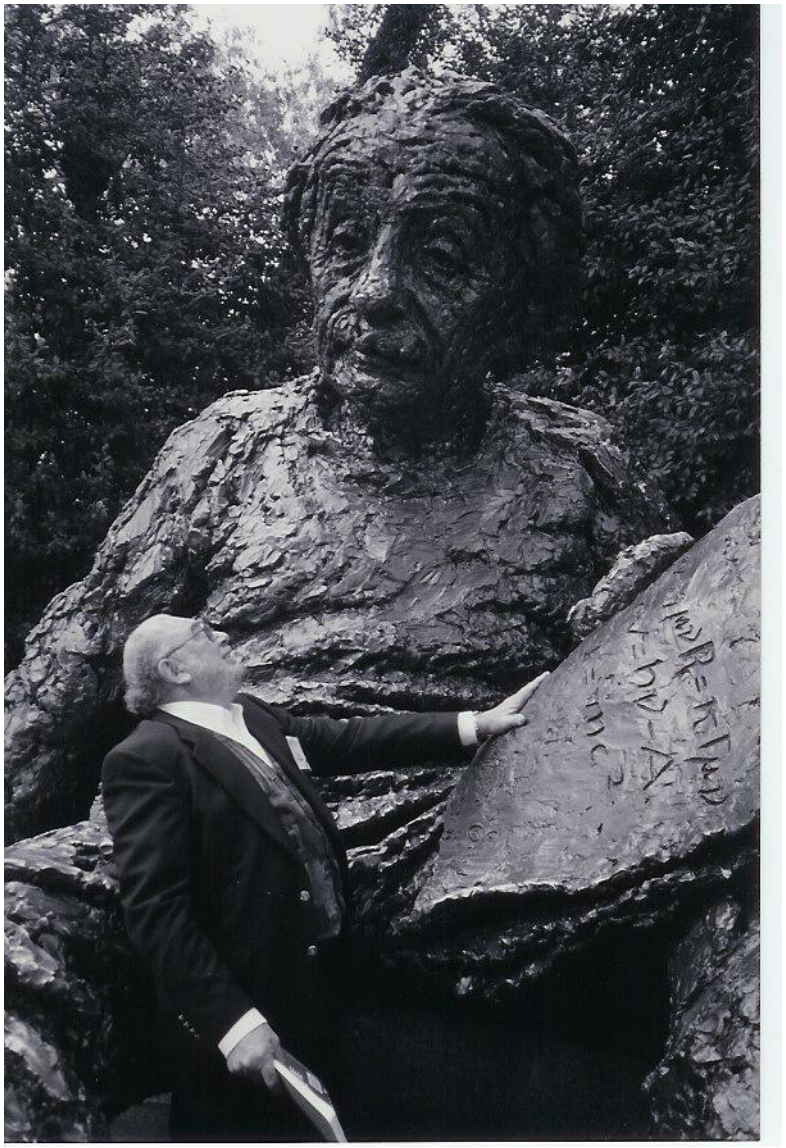
Adele Cutler

Utah State University

State-of-the-Art Data Mining Using Random Forests

Leo Breiman

January 27, 1928 - July 5, 2005



Outline

- What are random forests?
- How/why do they work?
- Useful for interpretation:
  - Out-of-bag data
  - Proximities
- Unbalanced data
- Unsupervised learning

CART
(Breiman, Friedman, Olshen, Stone 1984)

One of the most successful tools of the last 20 years.

Why?

1. Applicable for both classification and regression with no formal assumptions on the data structure.

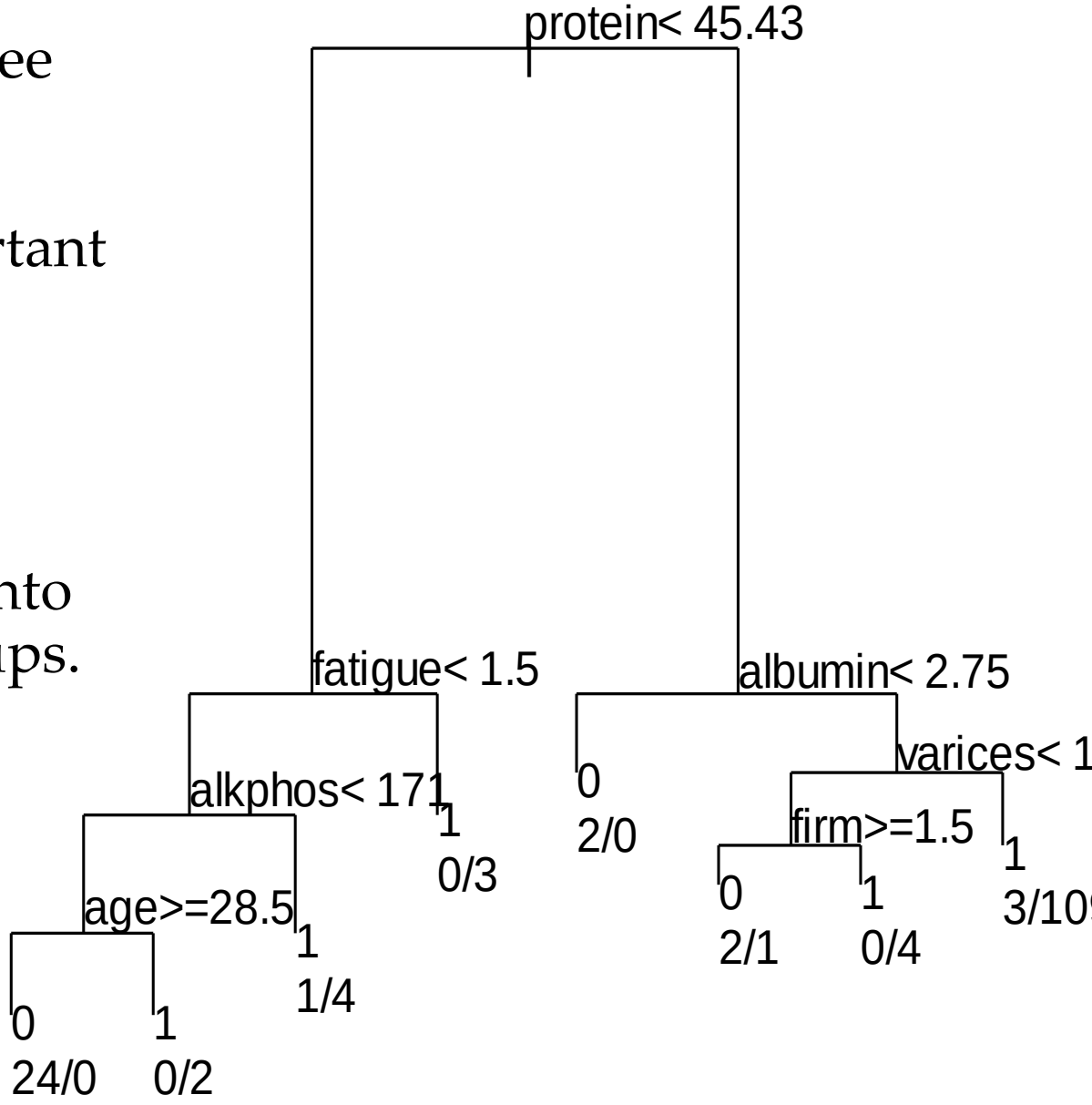
2. Can be applied to large datasets.

3. Handles missing data effectively.

4. Deals with categorical variables efficiently.

5. The picture of the tree can give valuable insights into which variables are important and where.

6. The terminal nodes suggest a natural clustering of data into homogeneous groups.



Drawbacks:

- **Accuracy**: state-of-the-art methods have much lower error rates than CART.

- **Instability**: if you change the data a little, the tree picture can change a lot, so the interpretation is built on shifting sands.

Random Forests: all the good things from CART **plus** state-of-the-art accuracy and stable interpretability.

What are Random Forests?

Grow a whole *forest* of trees:

- each tree is grown on an independent bootstrap sample* from the training data.
- independently, for each node of each tree, find the best split on ***m*** randomly selected variables.
- grow deep trees.

Get the prediction for a new case by voting (averaging) the predictions from all the trees.

*Sample N cases at random with replacement.

Properties of Random Forests

1. Accurate.
   - In independent tests on collections of data sets it's neck-and-neck with the best known machine learning methods (eg SVMs).

2. Fast.
   - With 100 variables, 100 trees in a forest can be grown in the same time as growing 3 single CART trees.

3. Does not overfit as we add more trees.

Properties (ctnd)

4. Handles
   - thousands of variables
   - many-valued categoricals
   - extensive missing values
   - badly unbalanced data sets.

5. Gives an internal estimate of test set error as trees are added to the ensemble.

6. Gives variable importance measures and proximities for visualization/clustering.

Gives a wealth of scientifically important insights!

Outline

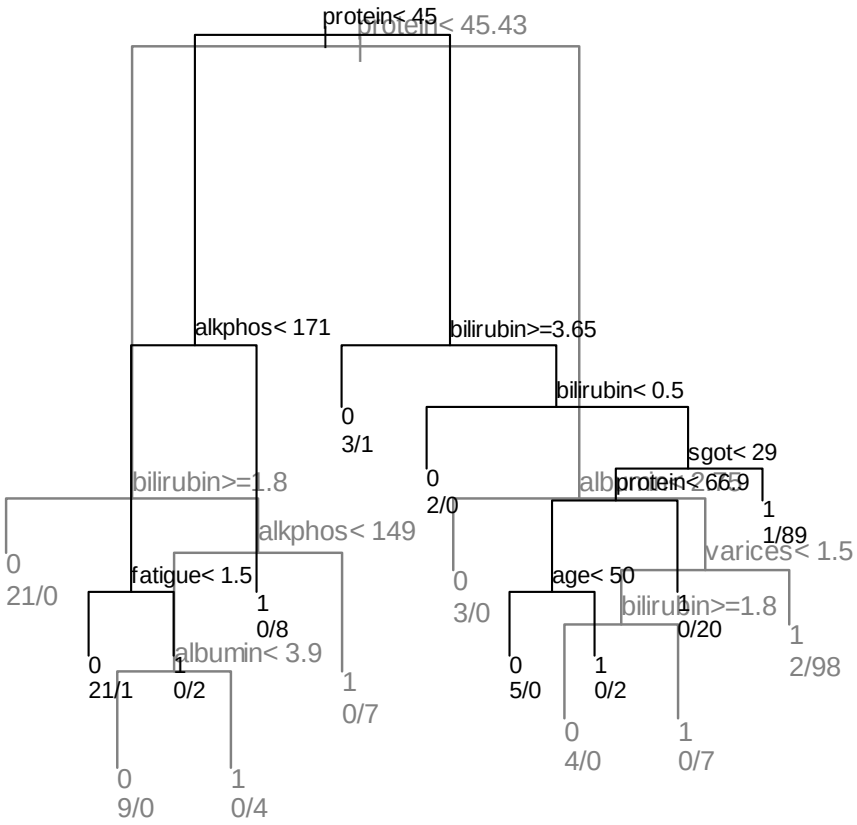
- What are random forests?
- How/why do they work?
- Useful for interpretation:
  - Out-of-bag data
  - Proximities
- Unbalanced data
- Unsupervised learning

Random Forests

How does it work?

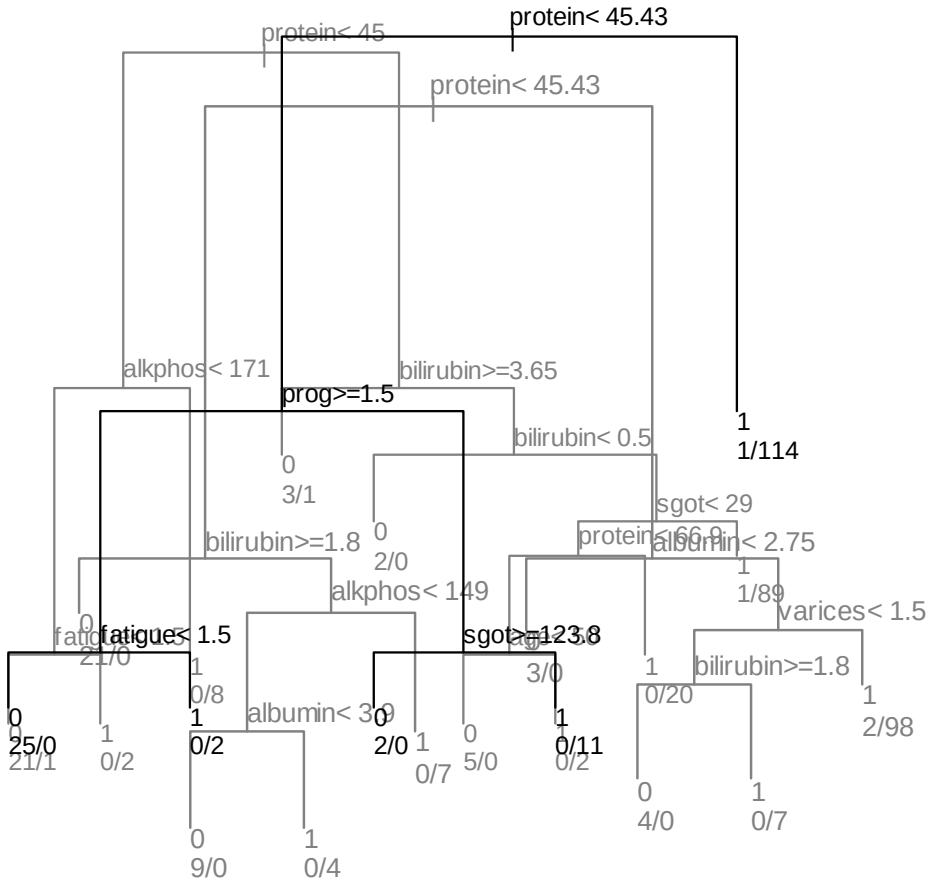
Random Forests

How does it work?



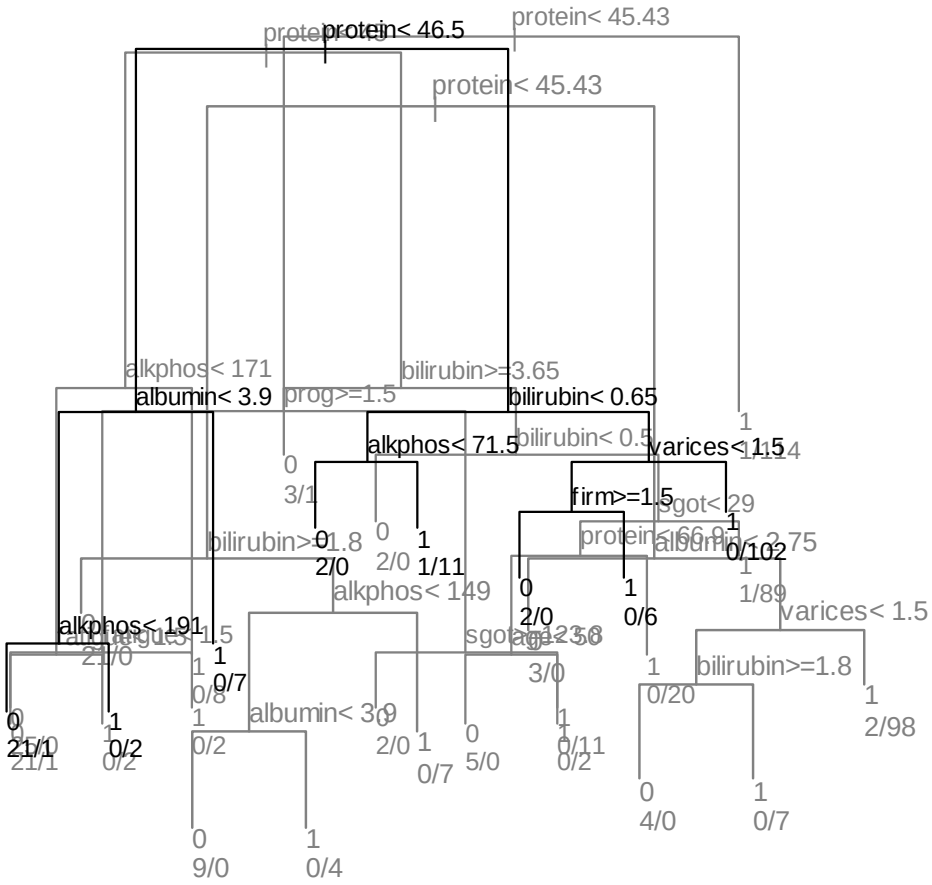
Random Forests

How does it work?



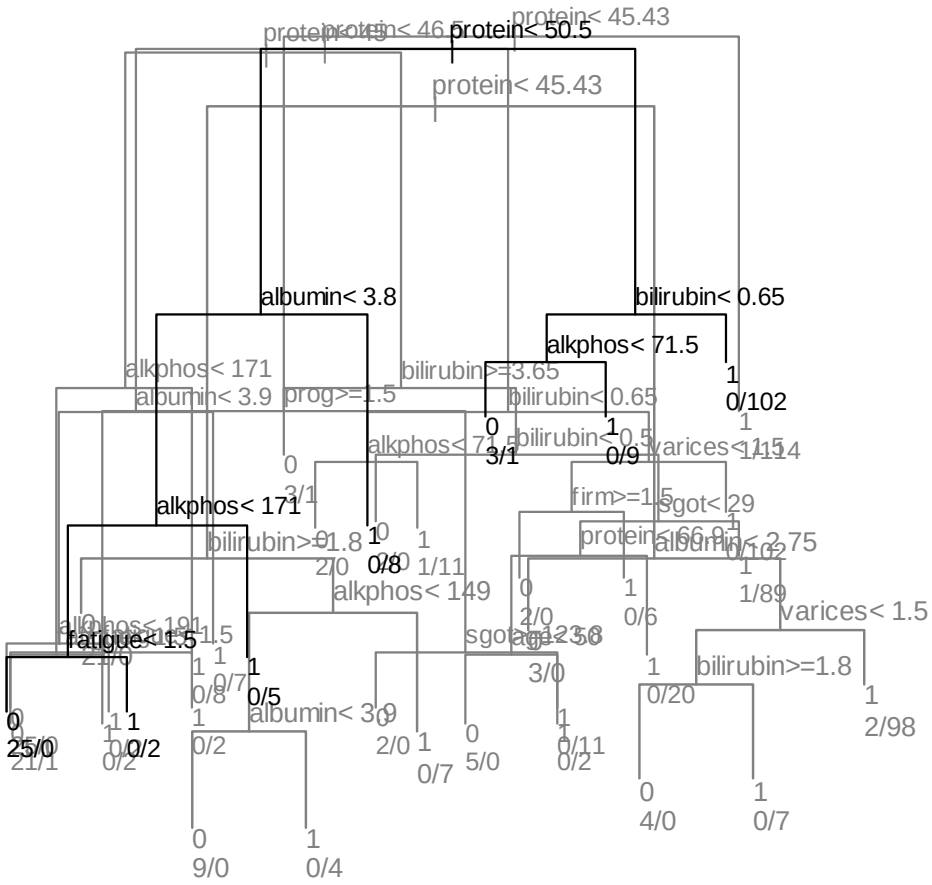
Random Forests

How does it work?



Random Forests

How does it work?



Random Forests

How does it work?

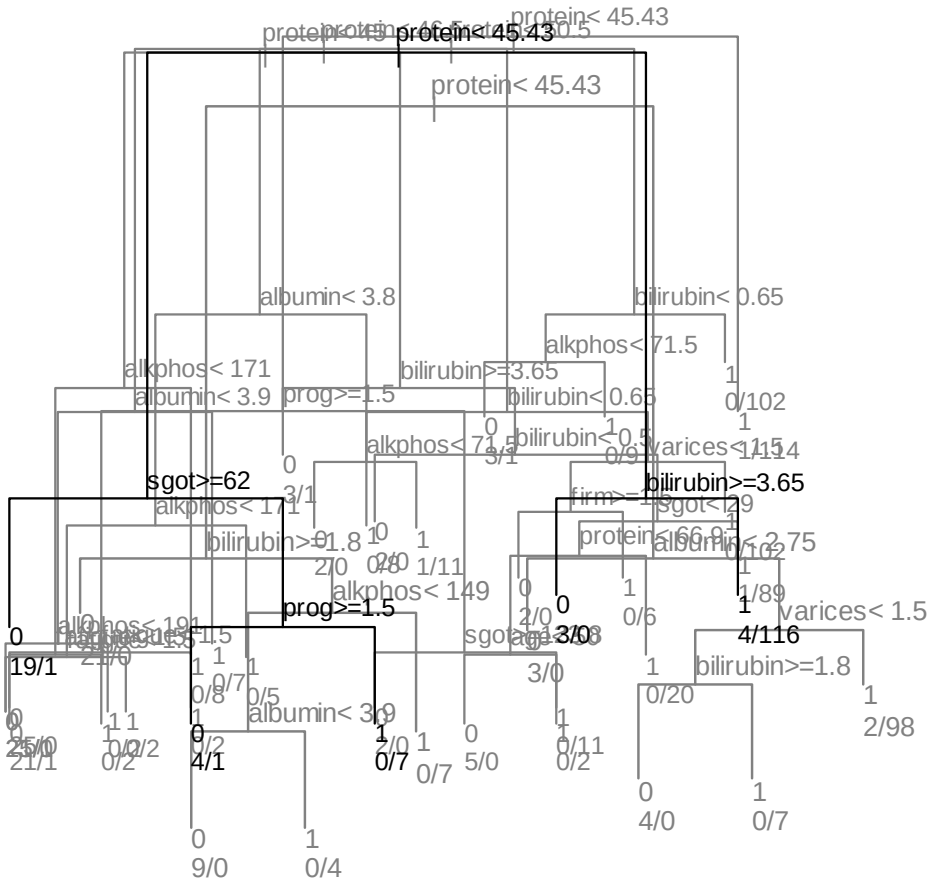


Illustration - Hepatitis

protein and alkaline phosphate

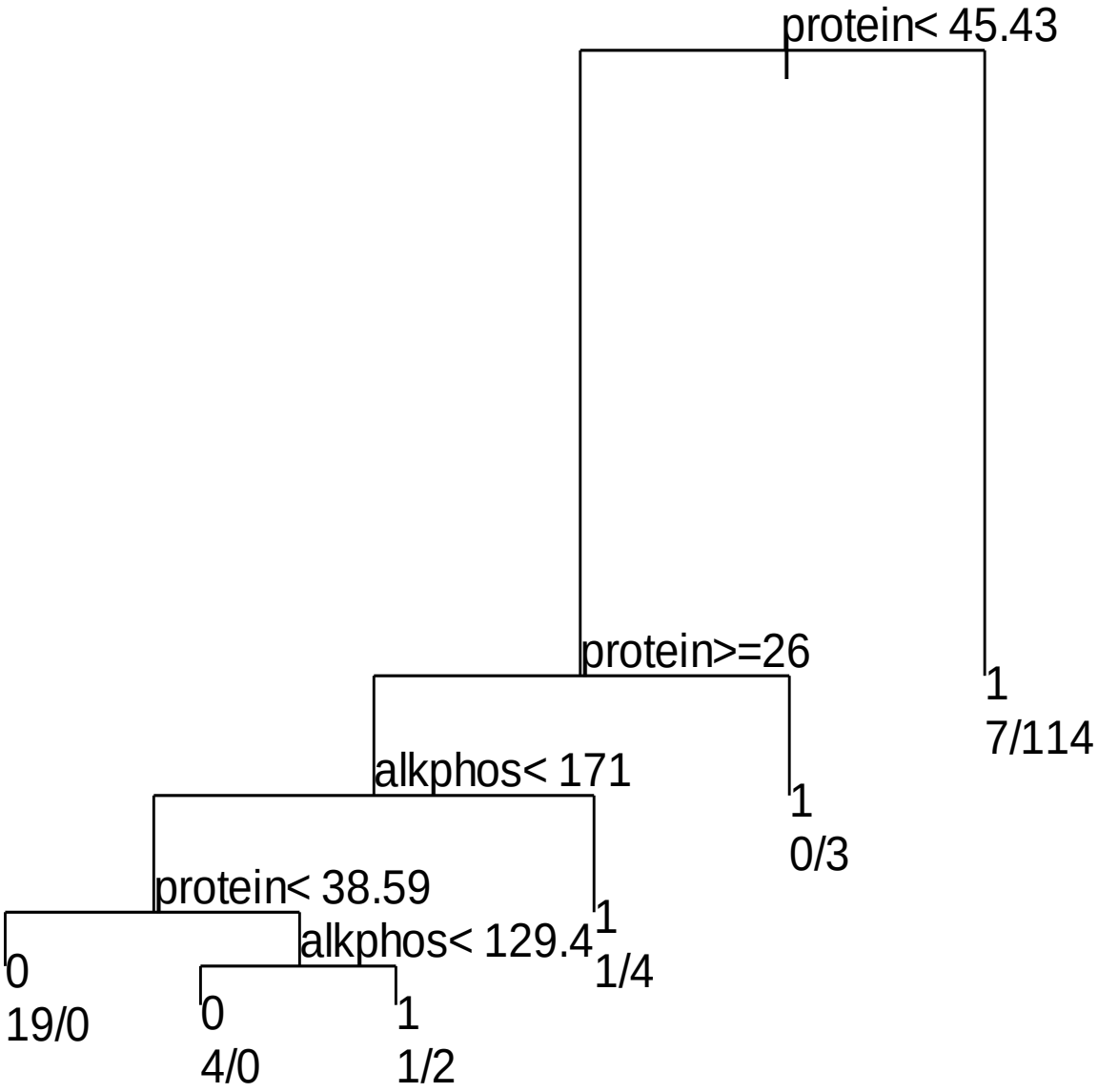
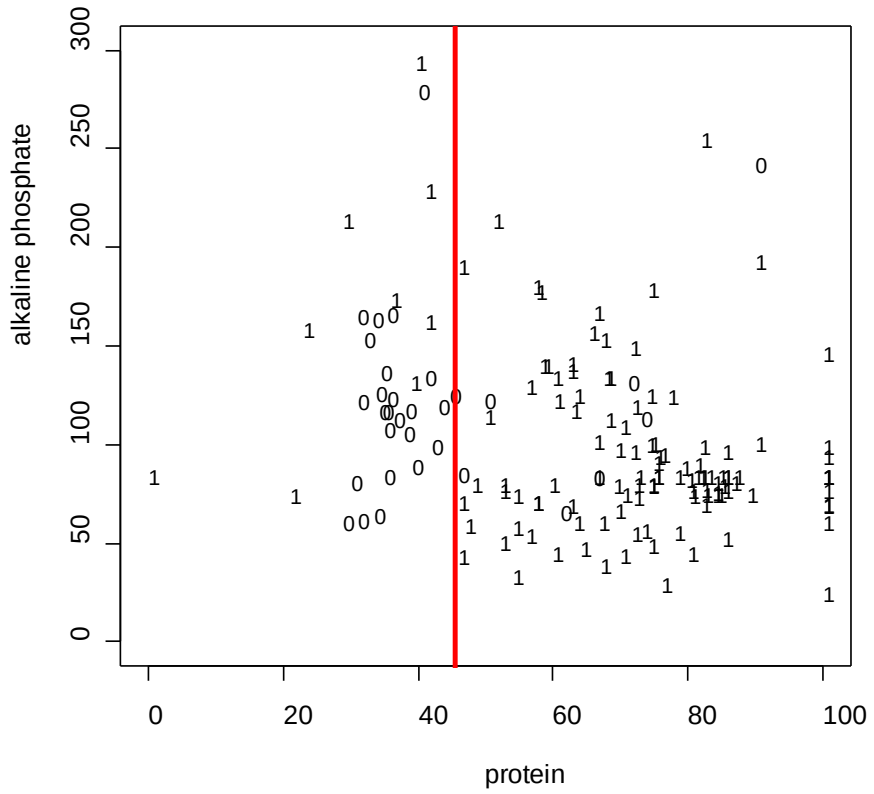


Illustration - Hepatitis

protein and alkaline phosphate



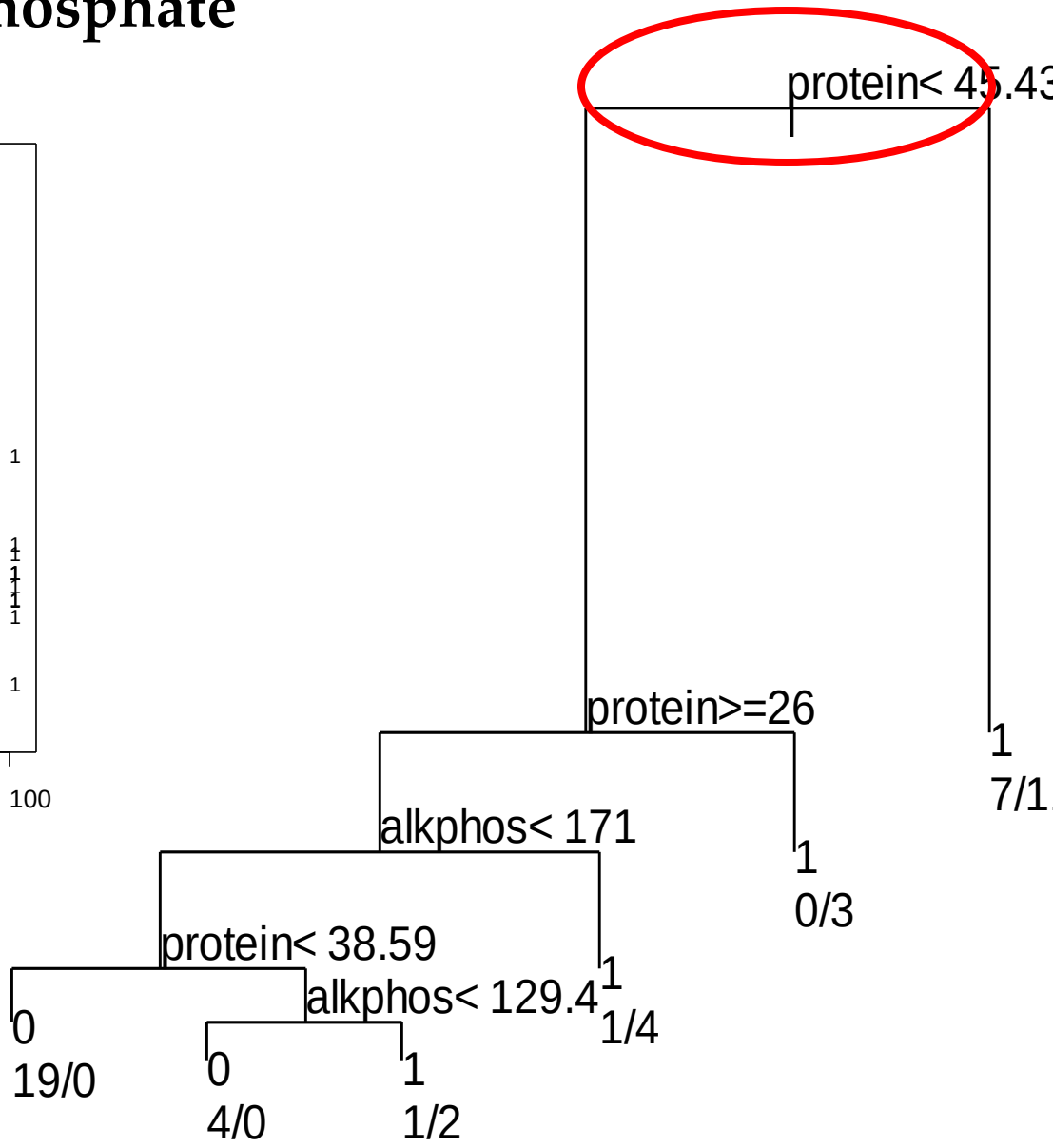
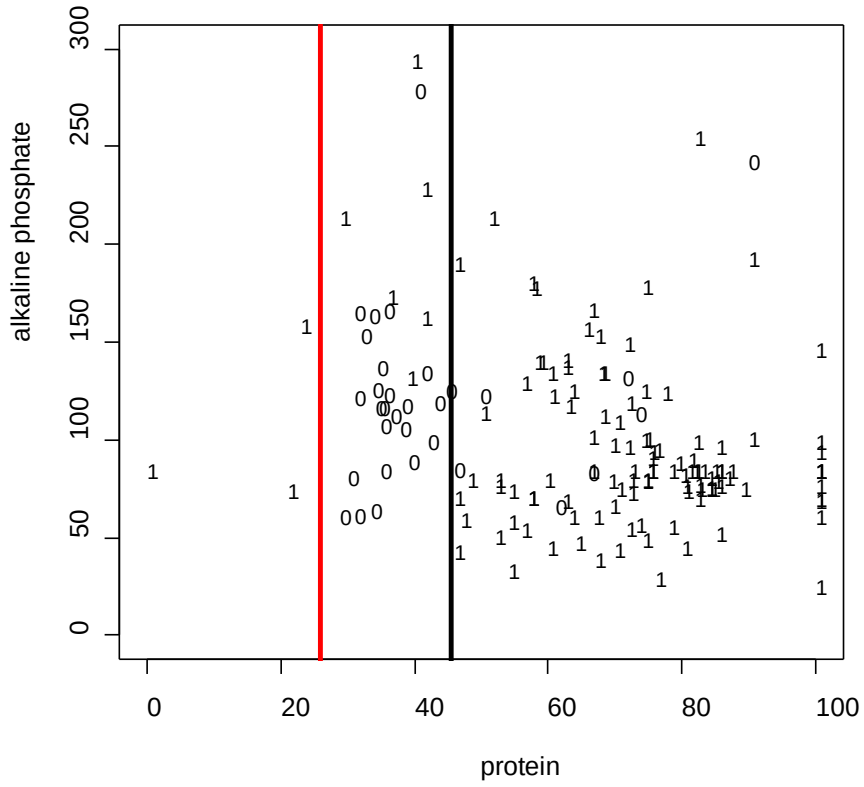


Illustration - Hepatitis

protein and alkaline phosphate



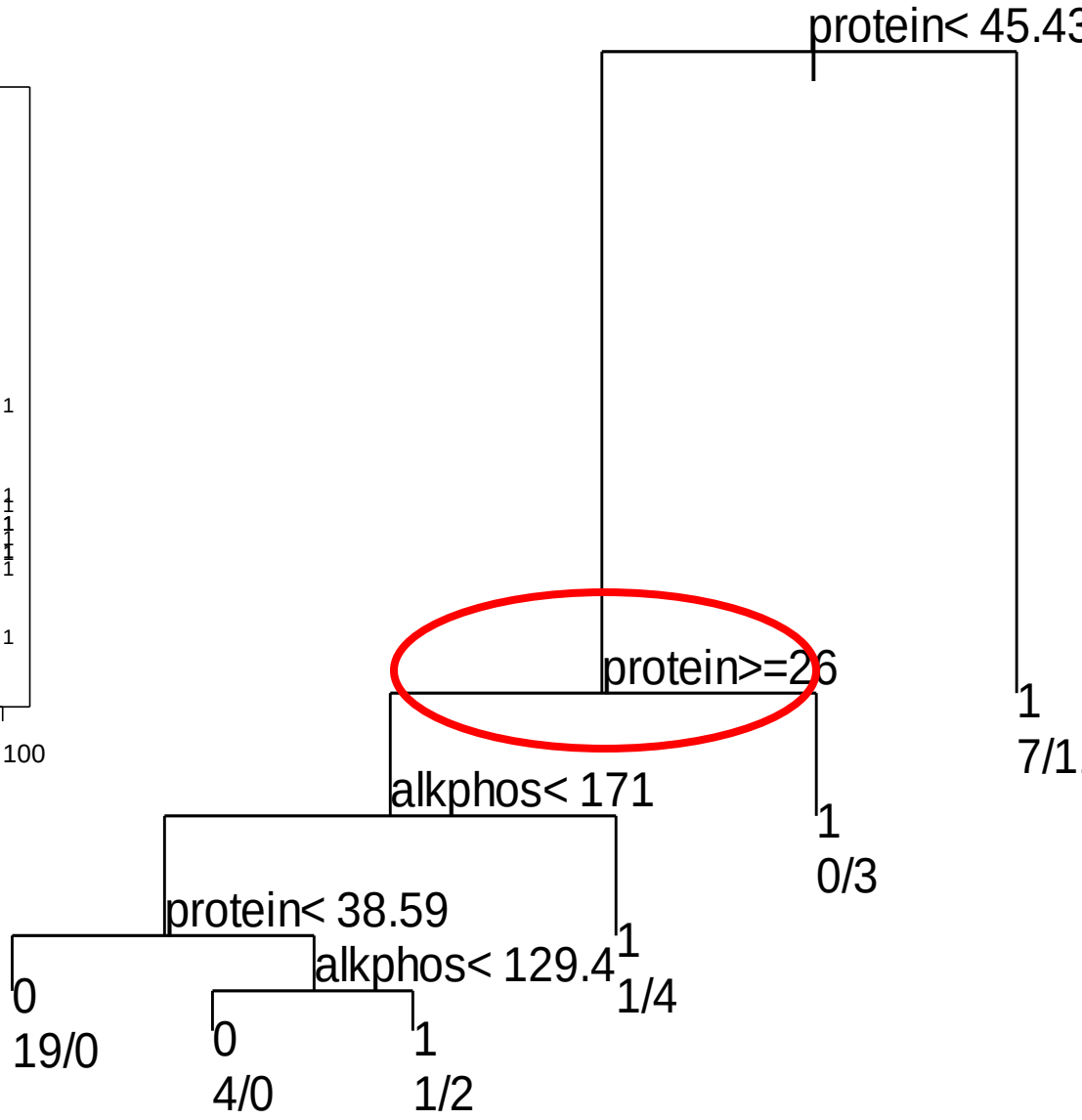
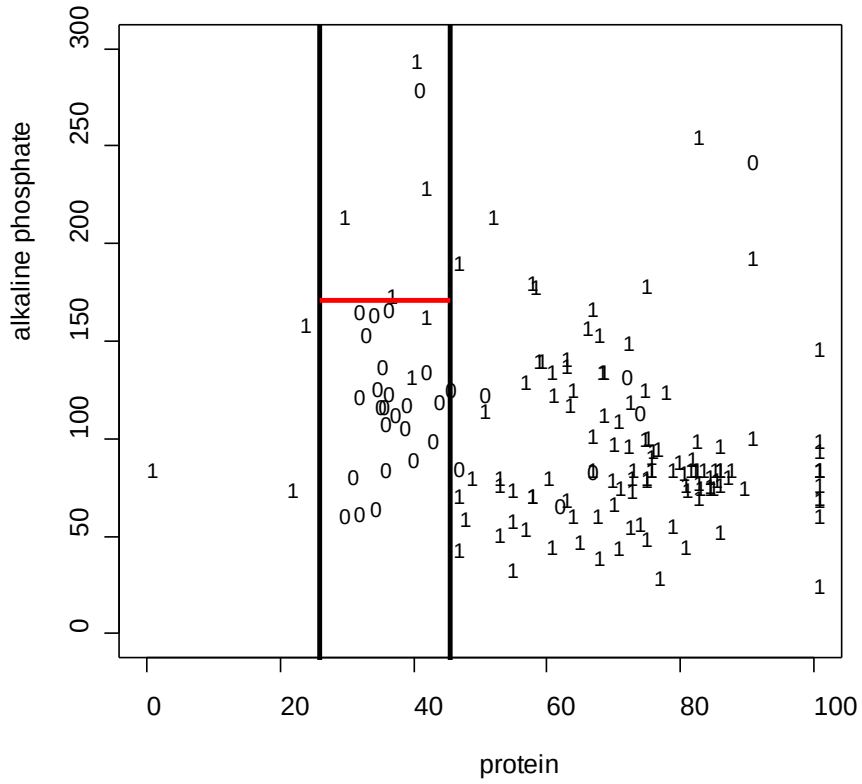


Illustration - Hepatitis

protein and alkaline phosphate



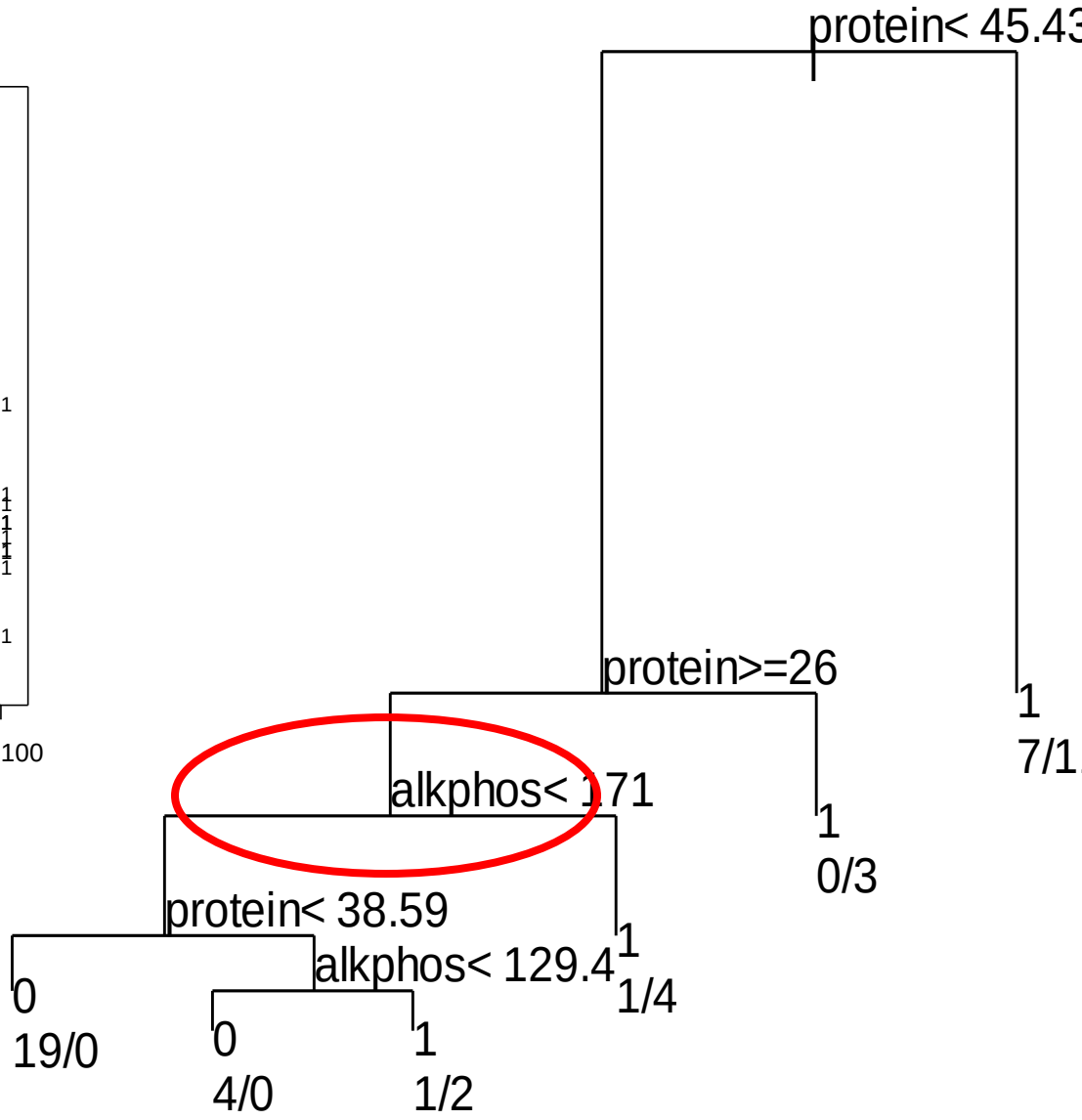
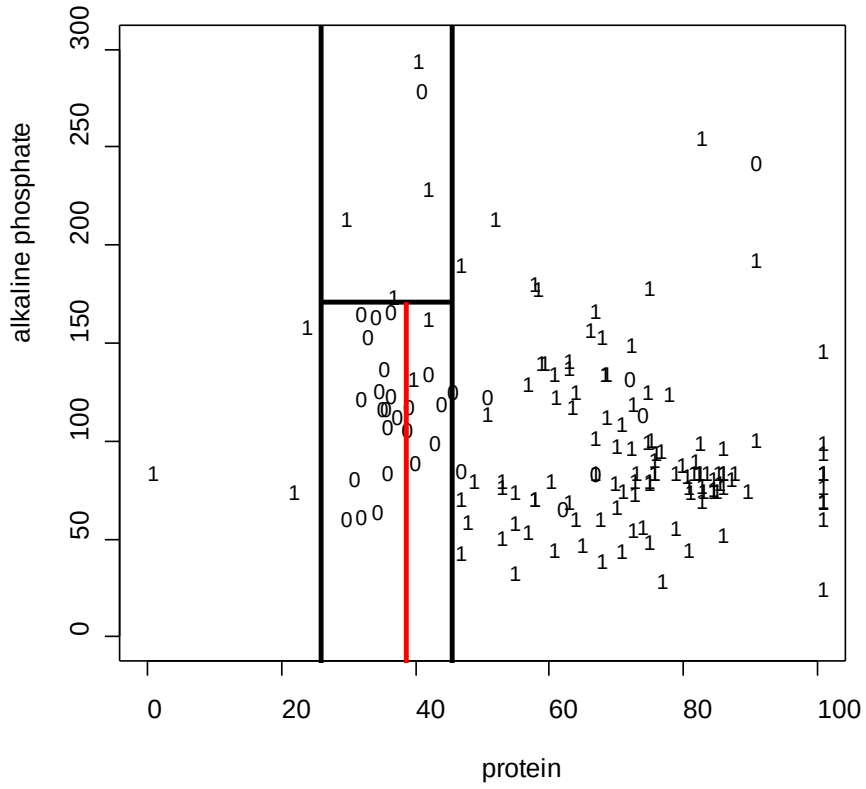


Illustration - Hepatitis

protein and alkaline phosphate



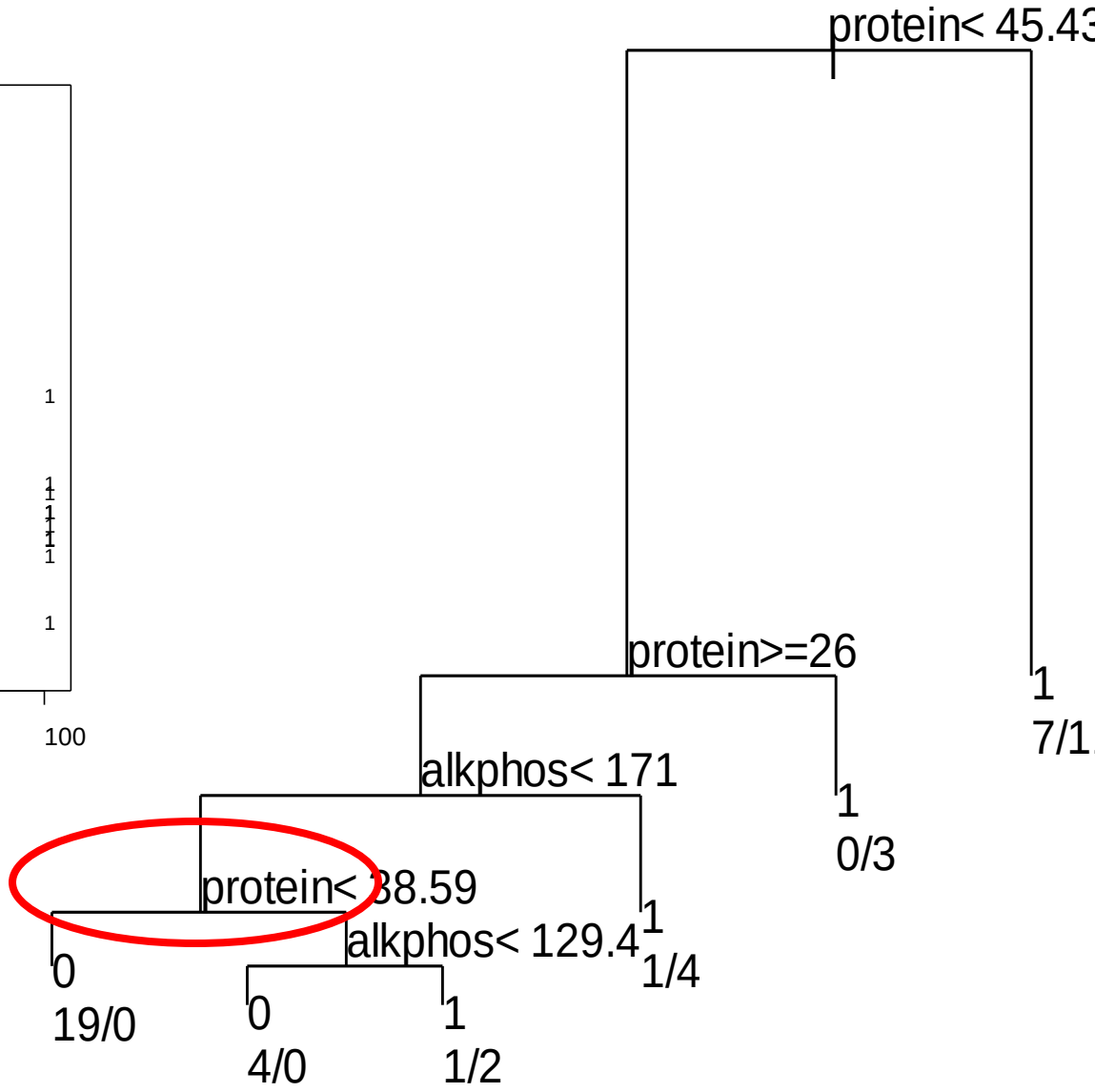
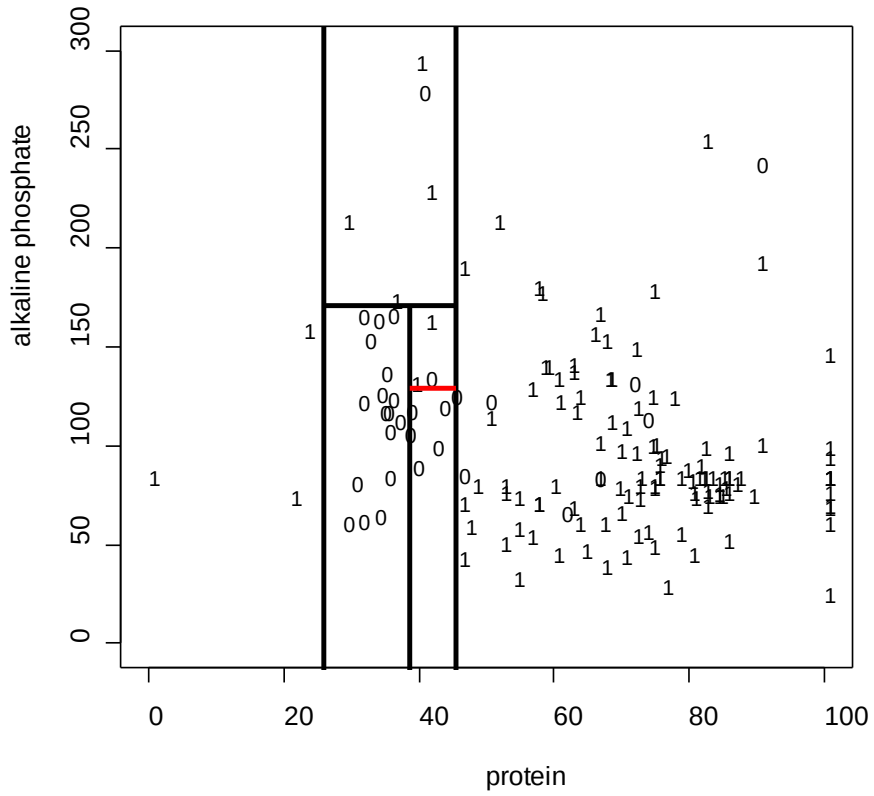
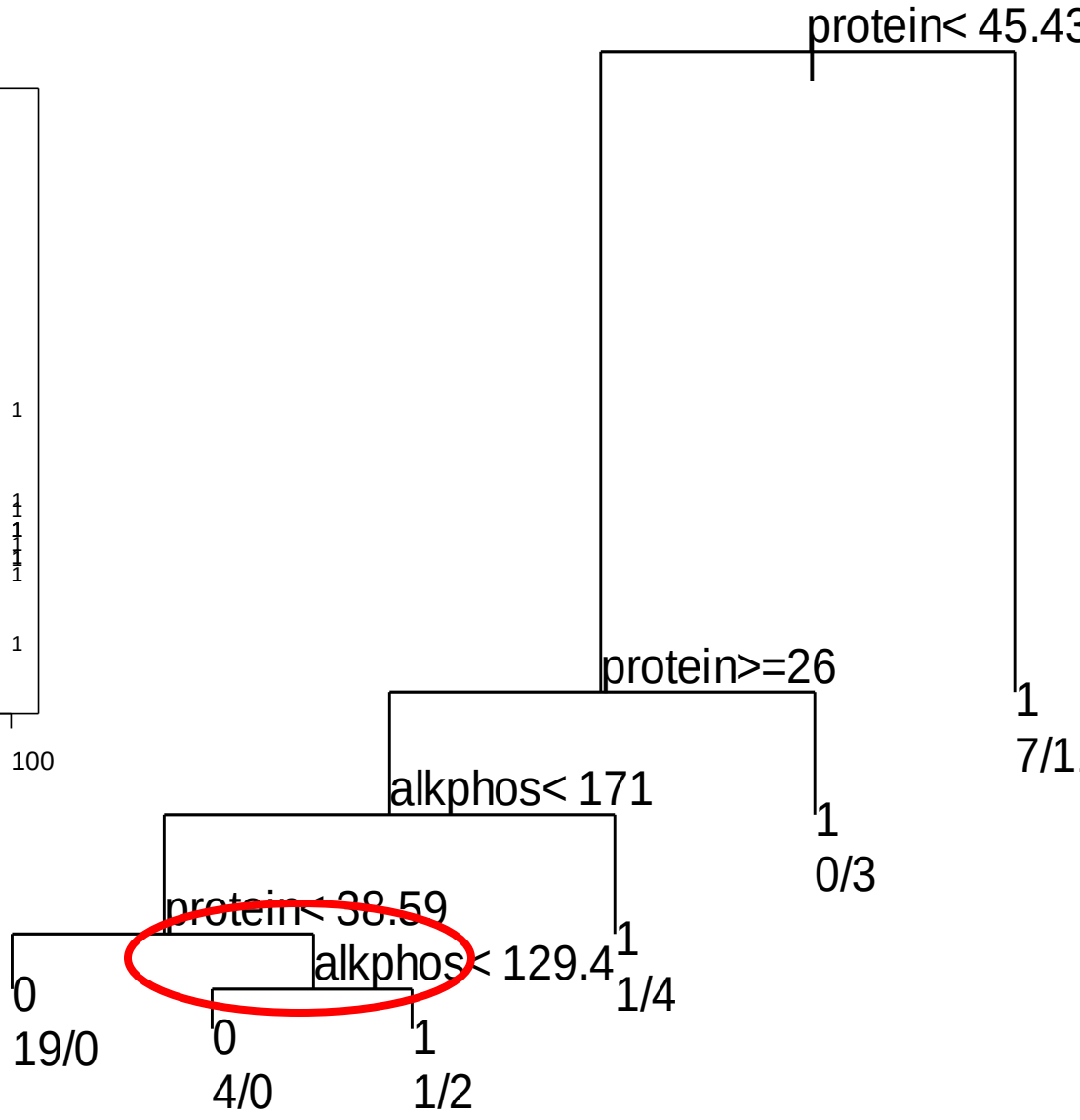
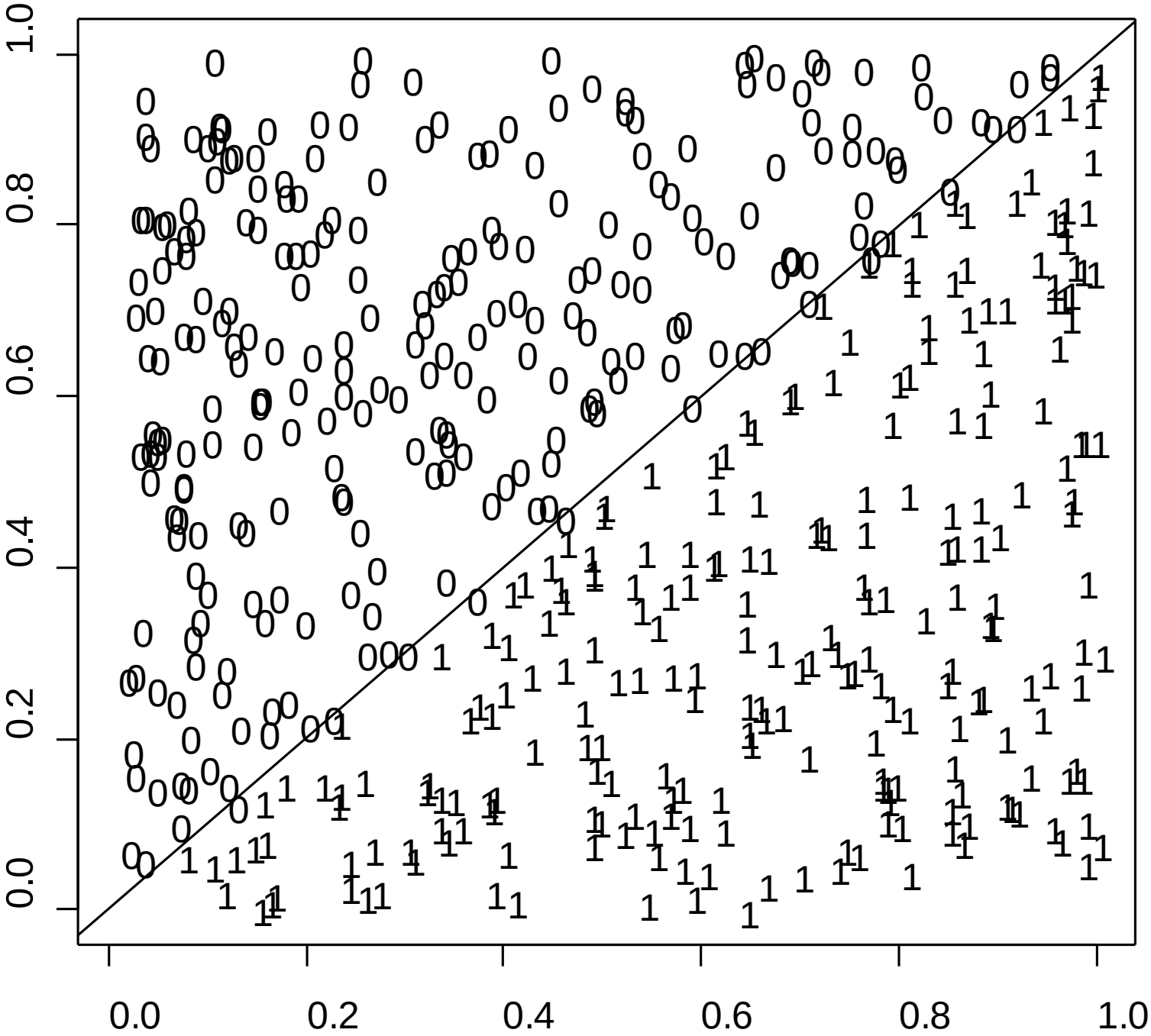


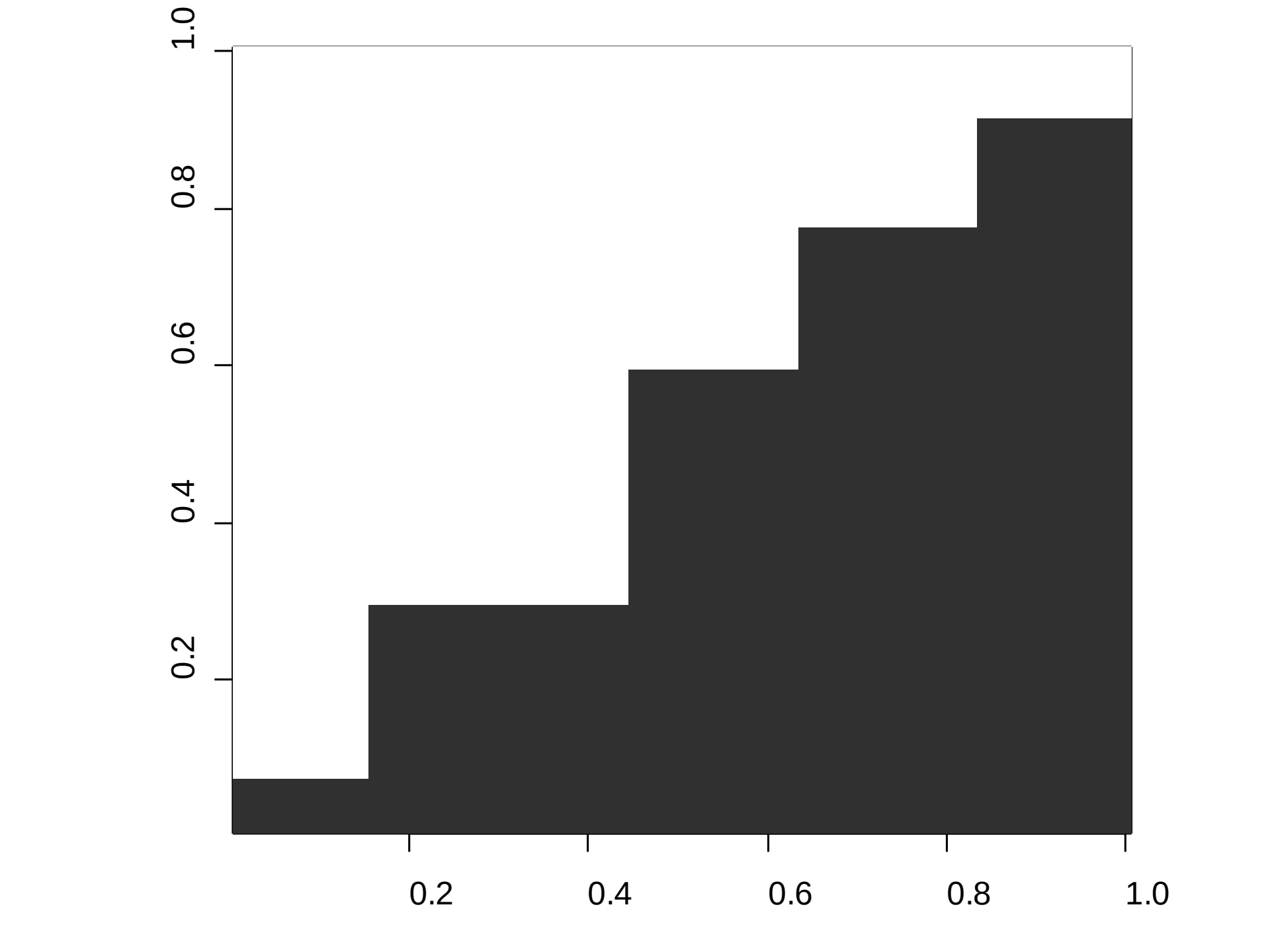
Illustration - Hepatitis

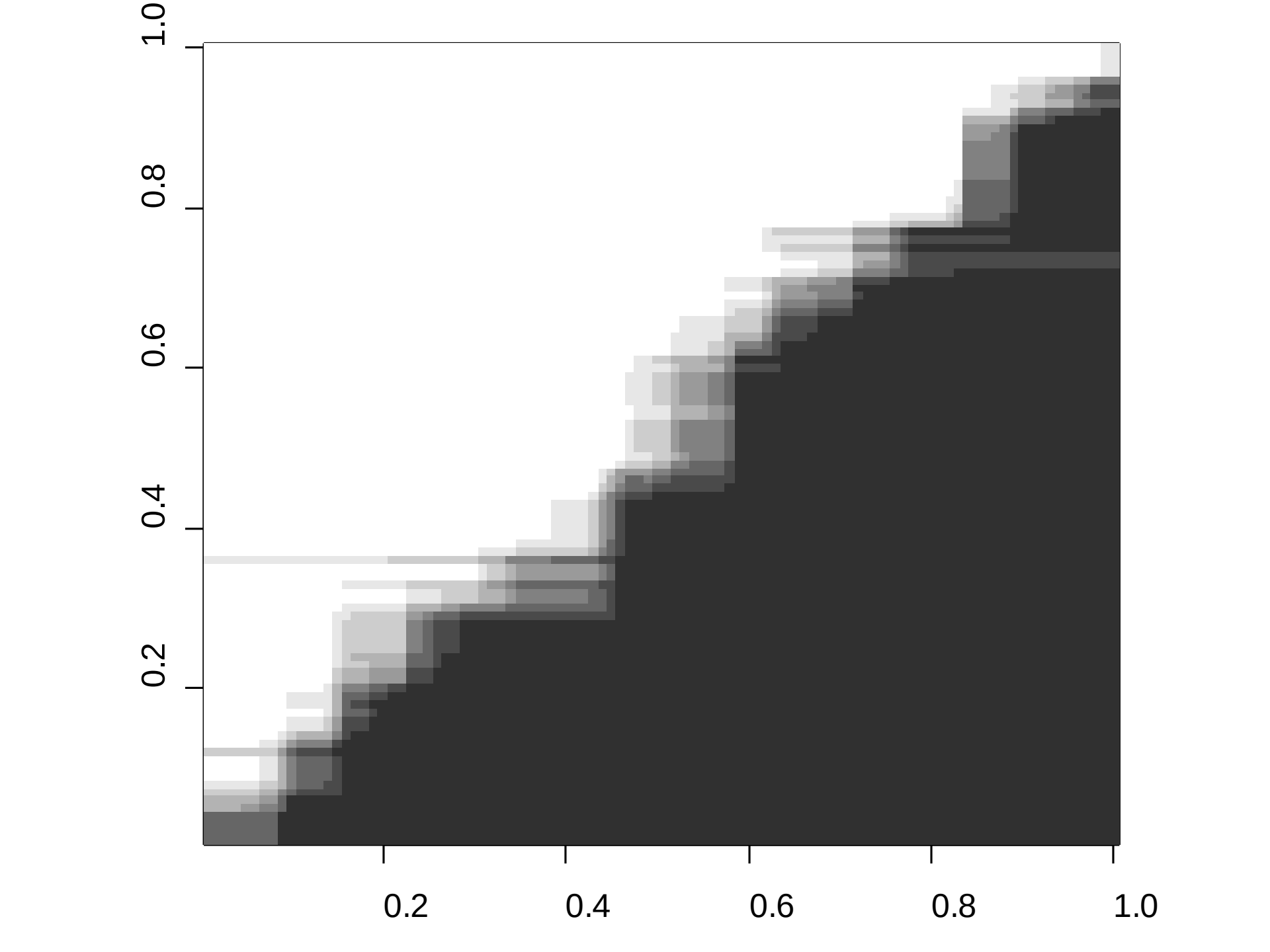
protein and alkaline phosphate

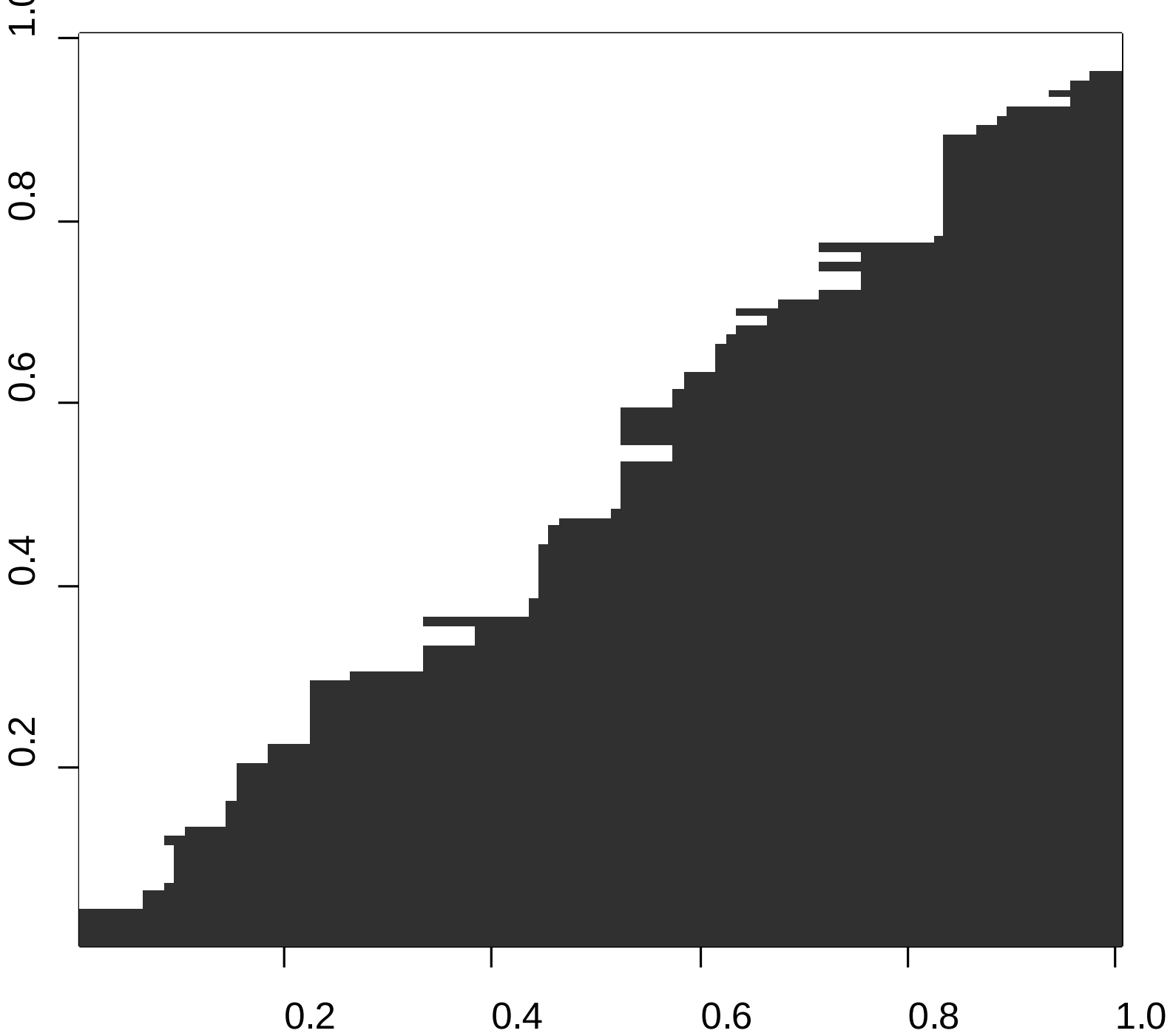




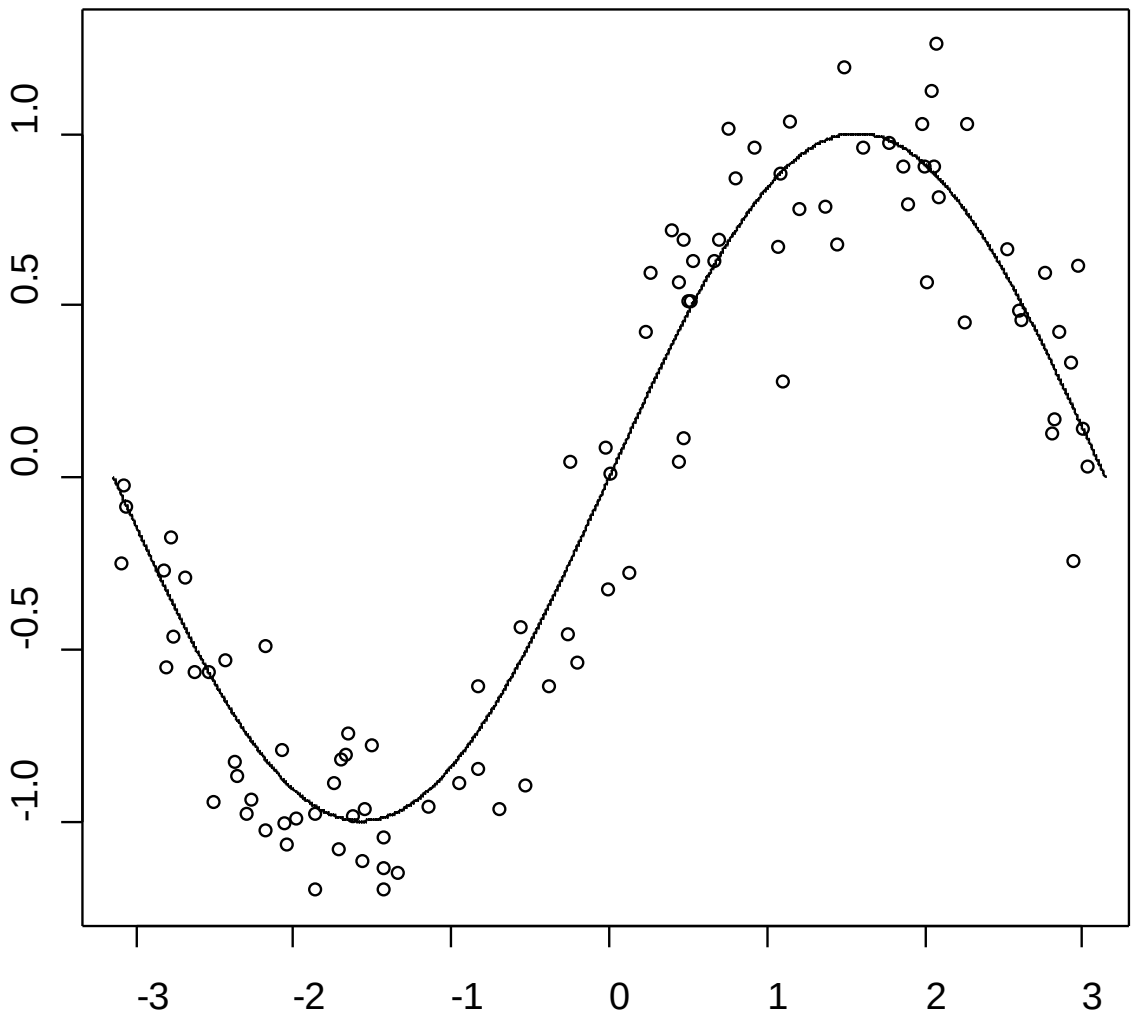

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0

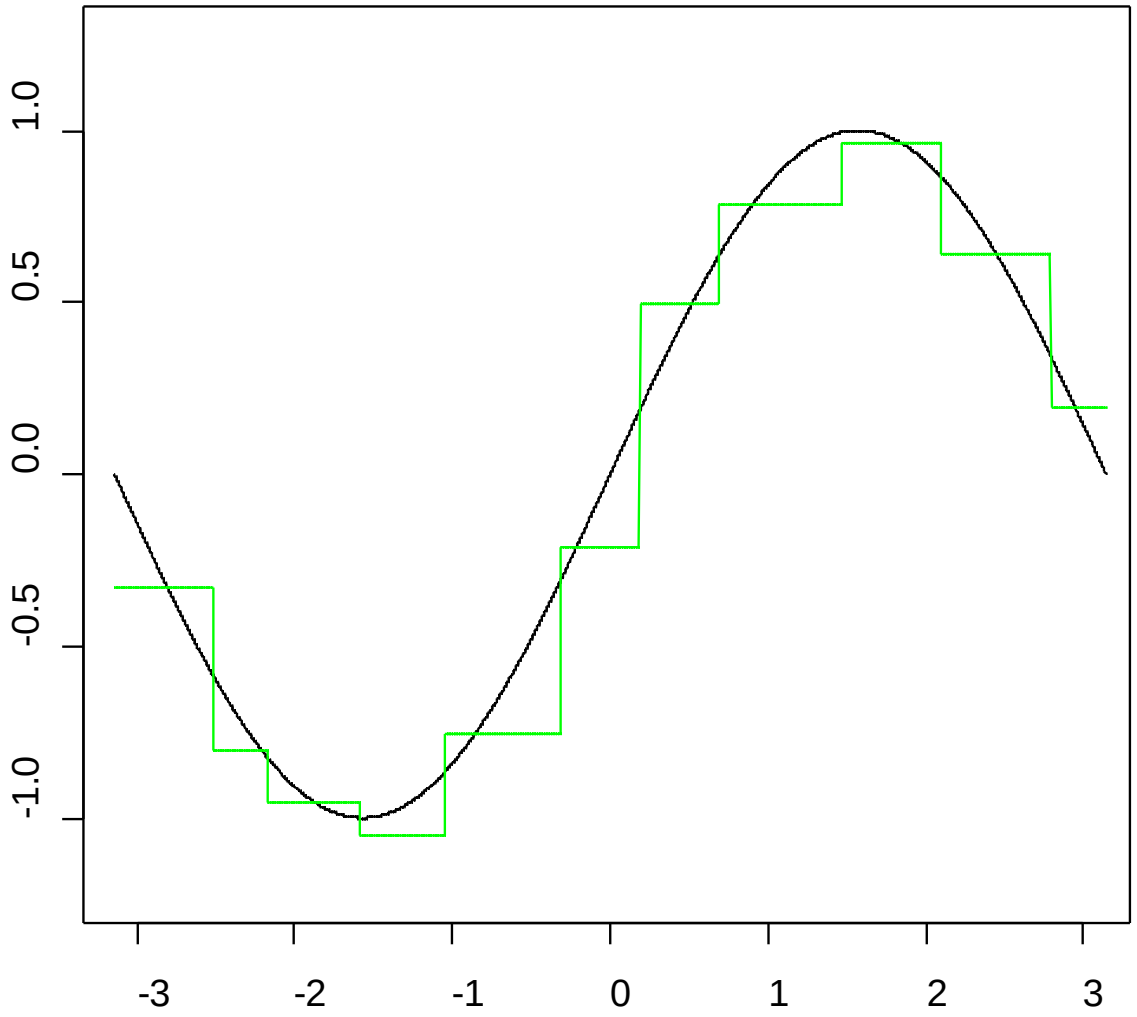

1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0


0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0

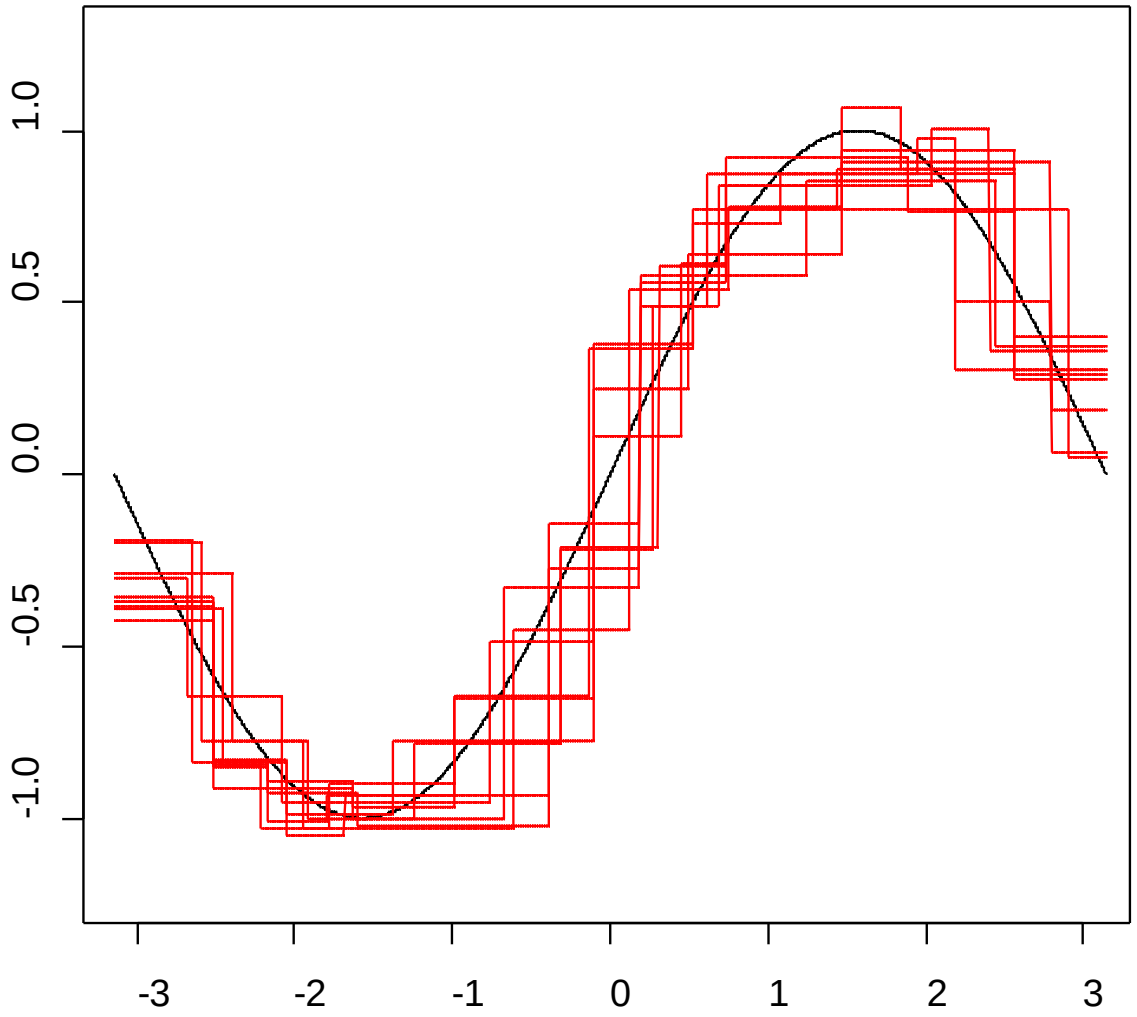
Data and Underlying Function



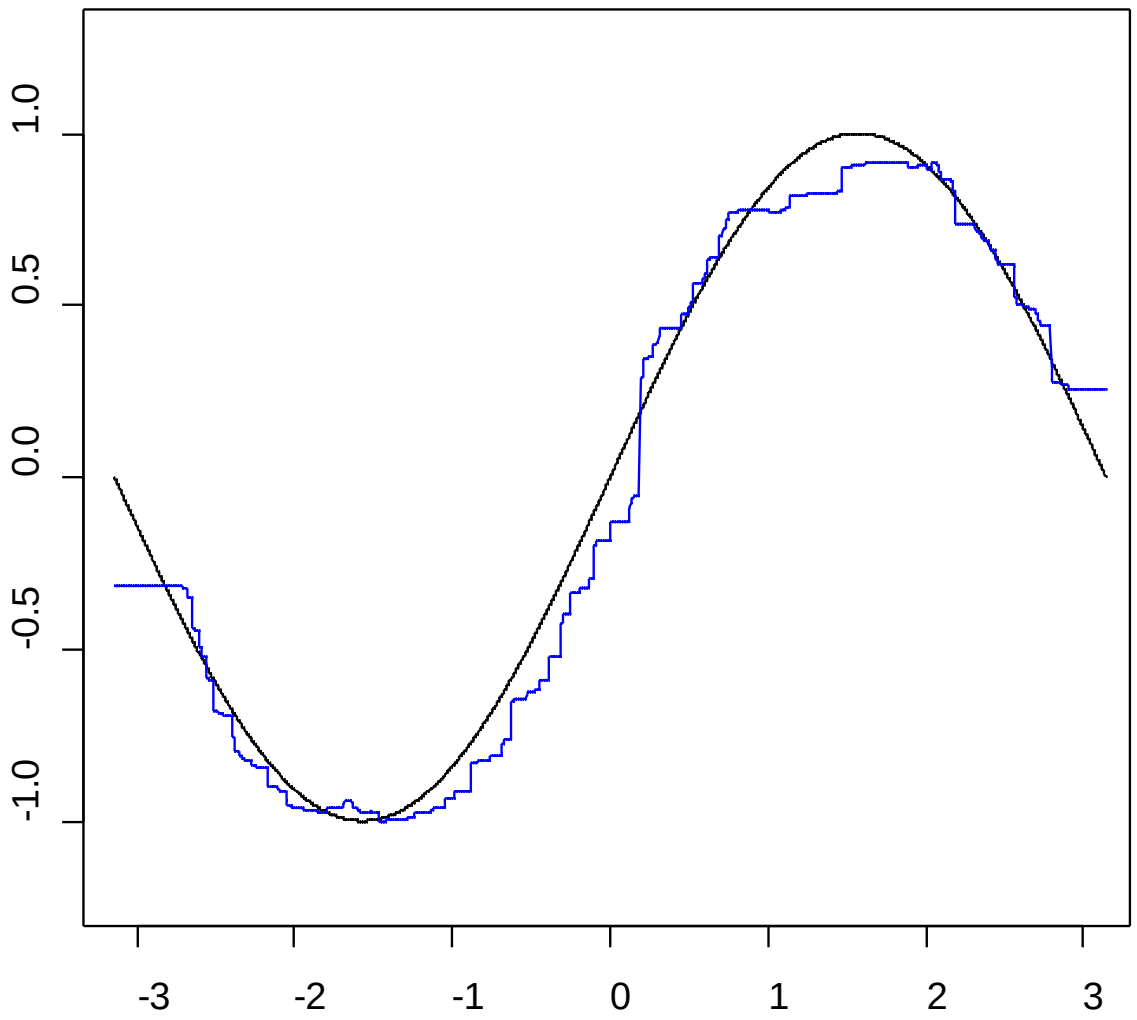
Single Regression Tree (all data)



10 Regression Trees (fit to boostrap samples)



Average of 100 Regression Trees (fit to bootstrap samples)



Random Forests in Context

An example of an *ensemble method:*

- Fit lots of classifiers
- Vote the predictions

Ensemble methods differ in how they fit the classifiers and what data they use.

- Bagging (Breiman)
- Boosting (Freund and Schapire)

Random Forests fits trees to bootstrap samples.

- WHY bootstrap? OUT-OF-BAG DATA
- WHY trees? PROXIMITIES

Random Forests in Context

An example of an *ensemble method:*

- Fit lots of classifiers
- Vote the predictions

Ensemble methods differ in how they fit the classifiers and what data they use.

- Bagging (Breiman)
- Boosting (Freund and Schapire)

Random Forests fits trees to bootstrap samples.

- WHY bootstrap? OUT-OF-BAG DATA
- WHY trees? PROXIMITIES

Out-of-bag Data

Think about a single tree from a random forest:

- The tree is grown on a bootstrap sample ("the bag").
- The remaining data are said to be "out-of-bag" (about one-third of the cases).
- The out-of-bag data serve as a test set for this tree.

Out-of-bag data give

- Estimated error rate
- A way to choose RF parameters (**m**, weights)
- Variable importance

The out-of-bag Error Rate

Think of a ***single case*** in the training set:

- It will be out-of-bag in about one-third of the trees.
- Predict its class for each of these trees.
- Its **RF prediction** is the most common predicted class.

For example, suppose we fit 1000 trees, and a case is out-of-bag in 339 of them, of which:

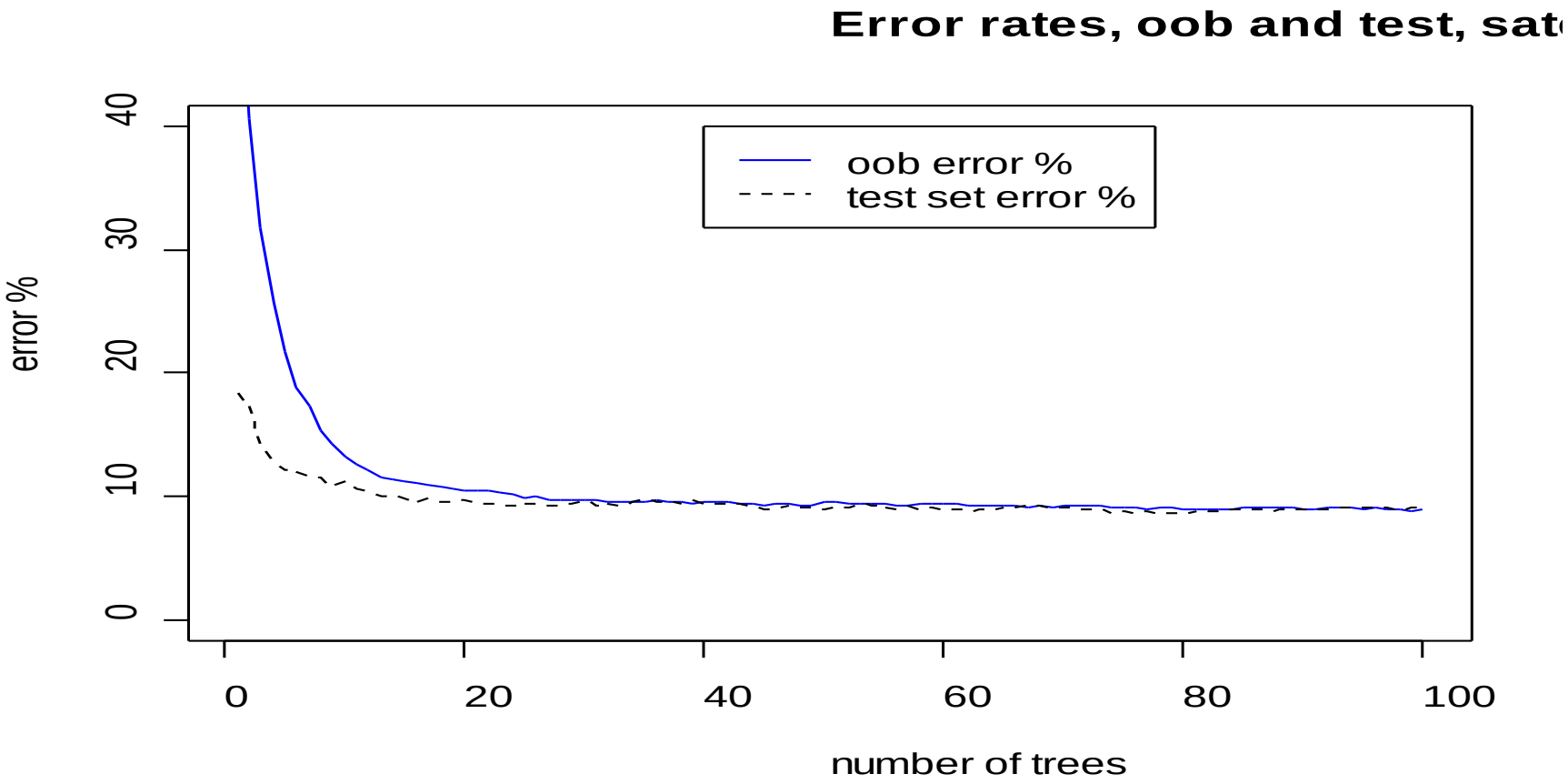
303 say "class 1"
36 say "class 2"

} The **RF prediction** is "class 1".

The out-of-bag error rate is simply the error rate of the RF predictor (can be done for each class).

Illustration – Satellite data

- 4435 cases, 36 variables.
- Test set: 2000 cases.



Using out-of-bag Data to Choose m

RF splits on the best of **m** randomly selected variables at each node. The out-of-bag error rate is used to select **m**.

Here's how:

1. Start with **m** = √**M**, **M**=total number of variables.
2. Run a few trees, recording the out-of-bag error rate.
3. Increase **m**, decrease **m**, until you are reasonably confident you've found a value with low out-of-bag error rate.

Variable Importance

For each tree, look at the out-of-bag data:

- randomly permute the values of variable *j*, holding the other variables fixed.
- pass these permuted data down the tree, save the classes.

Importance for variable *j* is

$$\left[\begin{array}{l}\text{error rate when}\\ \text{variable } j \text{ is permuted}\end{array}\right] - \left[\begin{array}{l}\text{error rate for the}\\ \text{original data}\end{array}\right]$$

where the first error rate is averaged over the out-of-bag data.

Illustration – Breast Cancer data

- 699 cases, 9 variables, two classes
- Initial out-of-bag error rate is 3.3%

Added 10,000 independent standard normals to each case, making 10,009 variables.

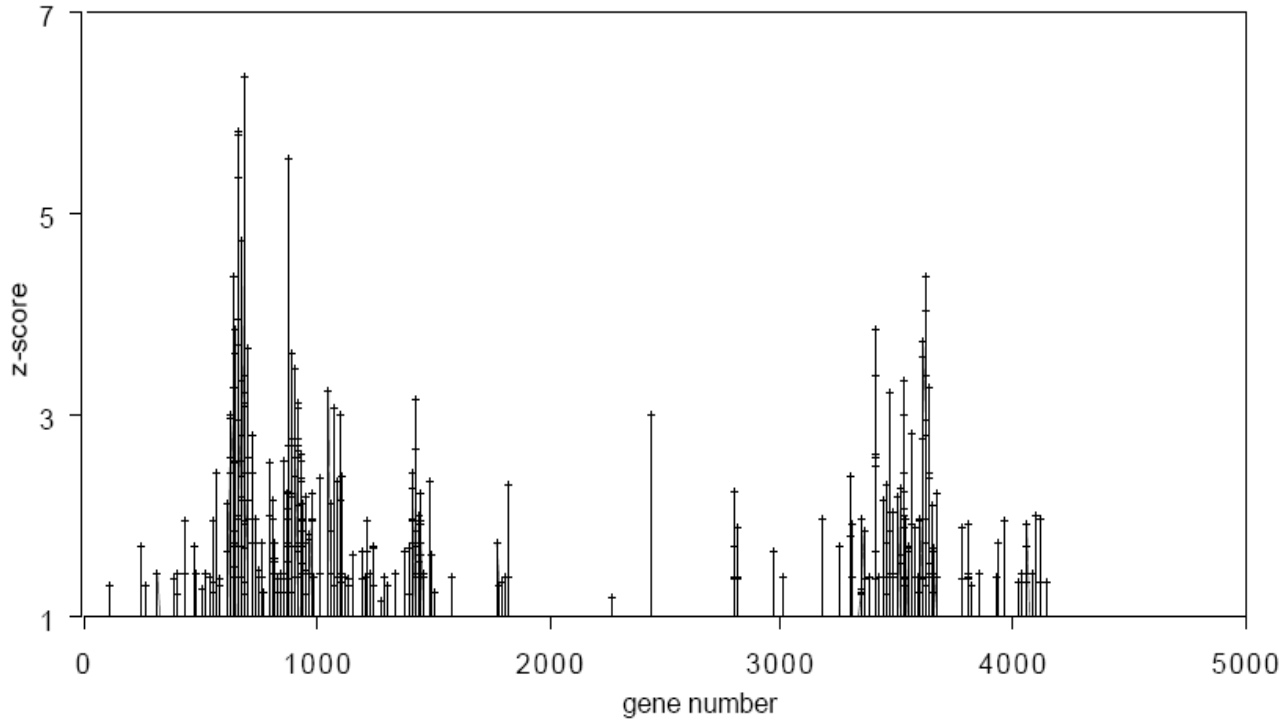
The twelve most important variables, chosen by RF:

variable #	raw score	z-score	significance
6	3.274	0.936	0.175
3	3.521	0.910	0.181
2	3.484	0.902	0.183
1	2.369	0.898	0.185
7	2.811	0.879	0.190
8	2.266	0.847	0.199
5	2.164	0.829	0.204
4	1.853	0.814	0.208
9	0.825	0.700	0.242
8104	0.016	0.204	0.419
430	0.005	0.155	0.438
5128	0.004	0.147	0.441

Illustration – Microarray data (lymphoma)

- 81 cases, 4682 variables, three classes.
- Initial out-of-bag error rate is 1.25% (one case).

Z-scores for 1000 trees:



Don’t take our word for it!

2003 NIPS competition on feature selection in high-dimensional data (thousands of variables):

- over 1600 entries from some of the most prominent people in machine learning
- top 2nd and 3rd entries used RF for feature selection.

Microarrays: March, 2005

Ramón Díaz-Uriarte, Sara Alvarez de Andrés

http://ligarto.org/rdiaz

Showed that RF gives good gene selection for microarray classification problems.

Random Forests in Context

An example of an *ensemble method:*

- Fit lots of classifiers
- Vote the predictions

Ensemble methods differ in how they fit the classifiers and what data they use.

- Bagging (Breiman)
- Boosting (Freund and Schapire)

Random Forests fits trees to bootstrap samples.

- WHY bootstrap? OUT-OF-BAG DATA
- WHY trees? PROXIMITIES

Proximities

Proximity of two items is the proportion of the time that they end up in the same node.

The proximities *don't* just measure similarity of the variables. They take into account whether the variable is important in distinguishing the classes.

- Two cases that have quite ***different*** variables might have ***large*** proximity if they differ only on variables that are ***not important***.

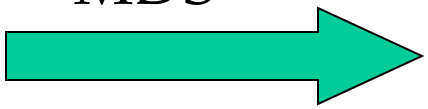
- Two cases that have quite ***similar*** inputs might have ***small*** proximity if they differ on inputs that are ***important.***

Proximities are used for

- Getting pictures
- Replacing missing values
- Detecting outliers

Getting Pictures with Scaling Variables

To "look" at the data we use classical multidimensional scaling (MDS) to get a picture in 2-D or 3-D:

Proximities 

scaling variables

Might see:

- clusters
- outliers
- other unusual structure.

Illustration – Microarray data

- 81 cases, 4682 variables, three classes.
- Initial out-of-bag error rate is 1.25% (one case).

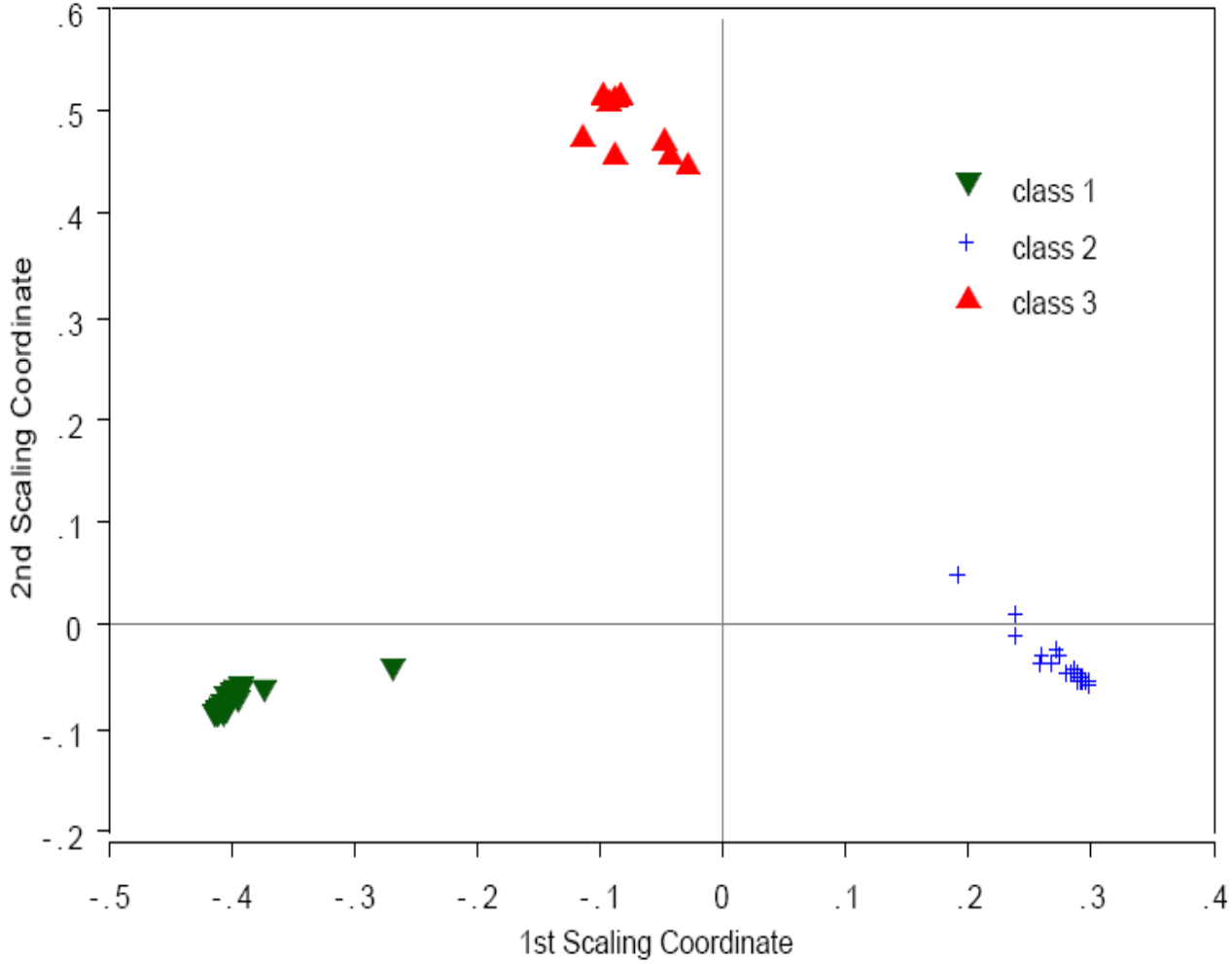
Other methods get similar results, but more is needed for the science:

1. What do the data look like? How do they cluster?
2. Which genes are active in discrimination?
3. What multivariate levels of gene expressions discriminate between classes?

The second question can be answered using variable importance. Let's take a look at 1 and 3.

Picturing the Microarray Data

The image below is a plot of the second scaling coordinate versus the first scaling coordinate.



Replacing Missing Values

Fast way: replace missing values for a given variable using the median of the non-missing values (or the most frequent, if categorical)

Better way (using proximities):

1. Start with the fast way.
2. Get proximities.
3. Replace missing values in case **n** by a weighted average of non-missing values, with weights proportional to the proximity between case **n** and the cases with the non-missing values.

Repeat steps 2 and 3 a few times (5 or 6).

Summary

Out-of-bag data give

- Estimated error rate
- Variable importance

Proximities give

- Illuminating pictures of the data
  - Clusters
  - Structure
  - Outliers
- Missing value fill-in
- Outlier detection

Outline

- What are random forests?
- How/why do they work?
- Useful for interpretation:
  - Out-of-bag data
  - Proximities
- Unbalanced data
- Unsupervised learning

Learning from Unbalanced Data

Increasingly often, data sets are occurring where the class of interest has a population that is a small fraction of the total population.

For such unbalanced data, a classifier can achieve great accuracy by classifying almost all cases into the majority class!

RF weights the classes to get similar error rates for each class.

Illustration – Satellite data

4435 cases, 36 variables, 2 classes.

Class 1 has 415 cases.

Run 1: ordinary run (RED)

Overall error rate	5.8%
Class 1 error rate	51.0%
Class 2 error rate	1.2%

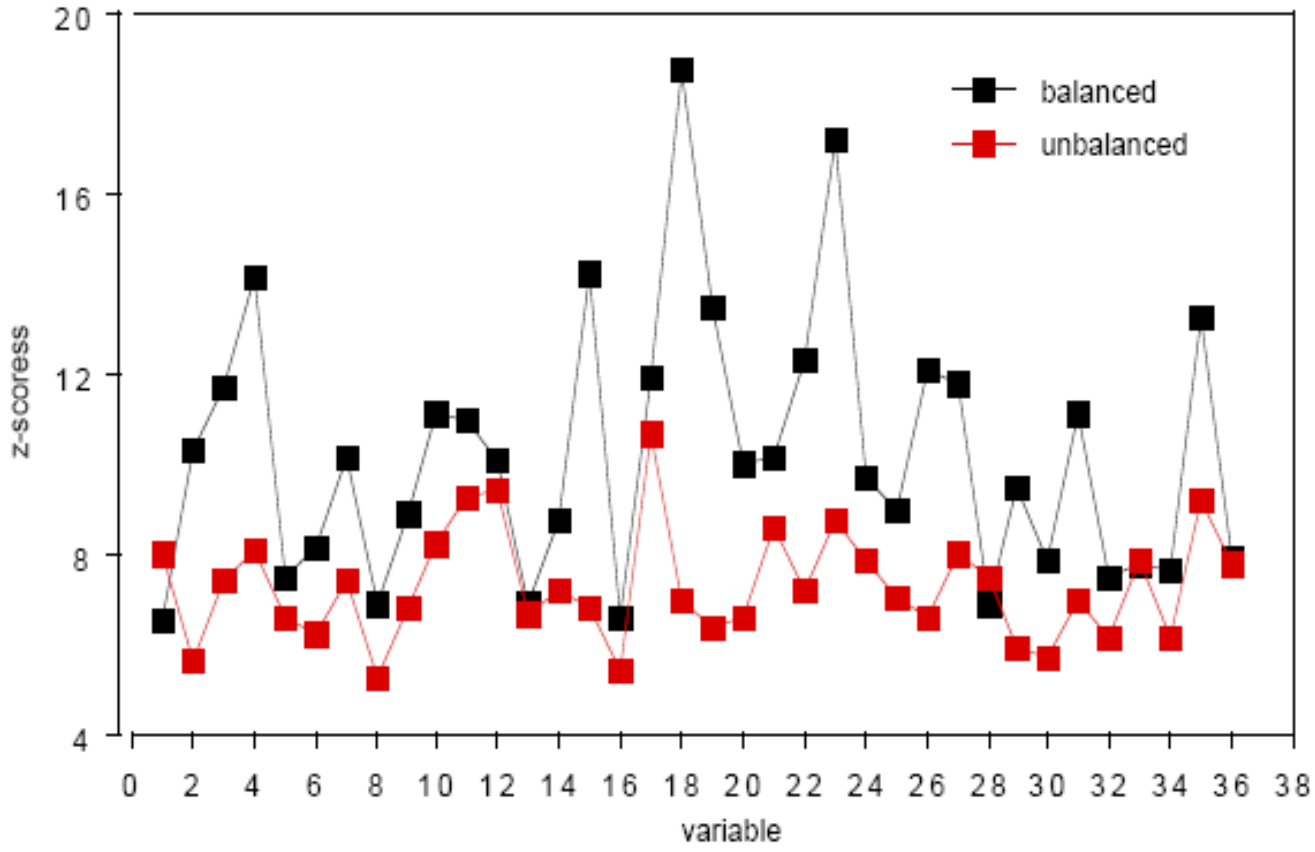
Run 2: Try to equalize the error rate for the two classes

Overall error rate	12.9%
Class 1 error rate	13.1%
Class 2 error rate	11.3%

8:1 weight on class 1

Variable Importances

Here is a graph of the z-scores for the two runs



Outline

- What are random forests?
- How/why do they work?
- Useful for interpretation:
  - Out-of-bag data
  - Proximities
- Unbalanced data
- Unsupervised learning

Unsupervised Learning

No class labels.

Most common approach: try to "cluster" the data to find some "structure".

- Ambiguous, ill-defined, no gold standard.
- Many methods rely on "white space" between "clusters" and are sensitive to outliers and noise.

However, it should be possible to determine structure, and see how it differs from noise.

The Unsupervised Learning Trick

Label the "real" data as class 1.

Construct cases from a synthetic second class as follows:

- Randomly select the value for variable 1 from all the observed values of variable 1.
- Independently, randomly select the value for variable 2 from all the observed values of variable 2.
- Similarly for variables 3,…,M.

The Synthetic Second Class

The synthetic second class has the same marginal distributions as the "real" data, but we have destroyed all the dependencies between the variables.

Now we have a 2-class classification problem.

Run random forests !

RF for Unsupervised Learning

If the out-of-bag misclassification rate is around 50%, random forests cannot distinguish between the classes.

This means that (according to random forests) the "real" data look like random sampling from **M** independent random variables.

But if the separation is good, then all the tools in random forests can be used to learn about the structure of the data…

A Proposed Test of Clustering Methods

- Consider real data with class labels.
- Erase the labels.
- Cluster the data.
- Do the clusters correspond to the original classes?

Let's try it!

Illustration – Microarray data (10% error)

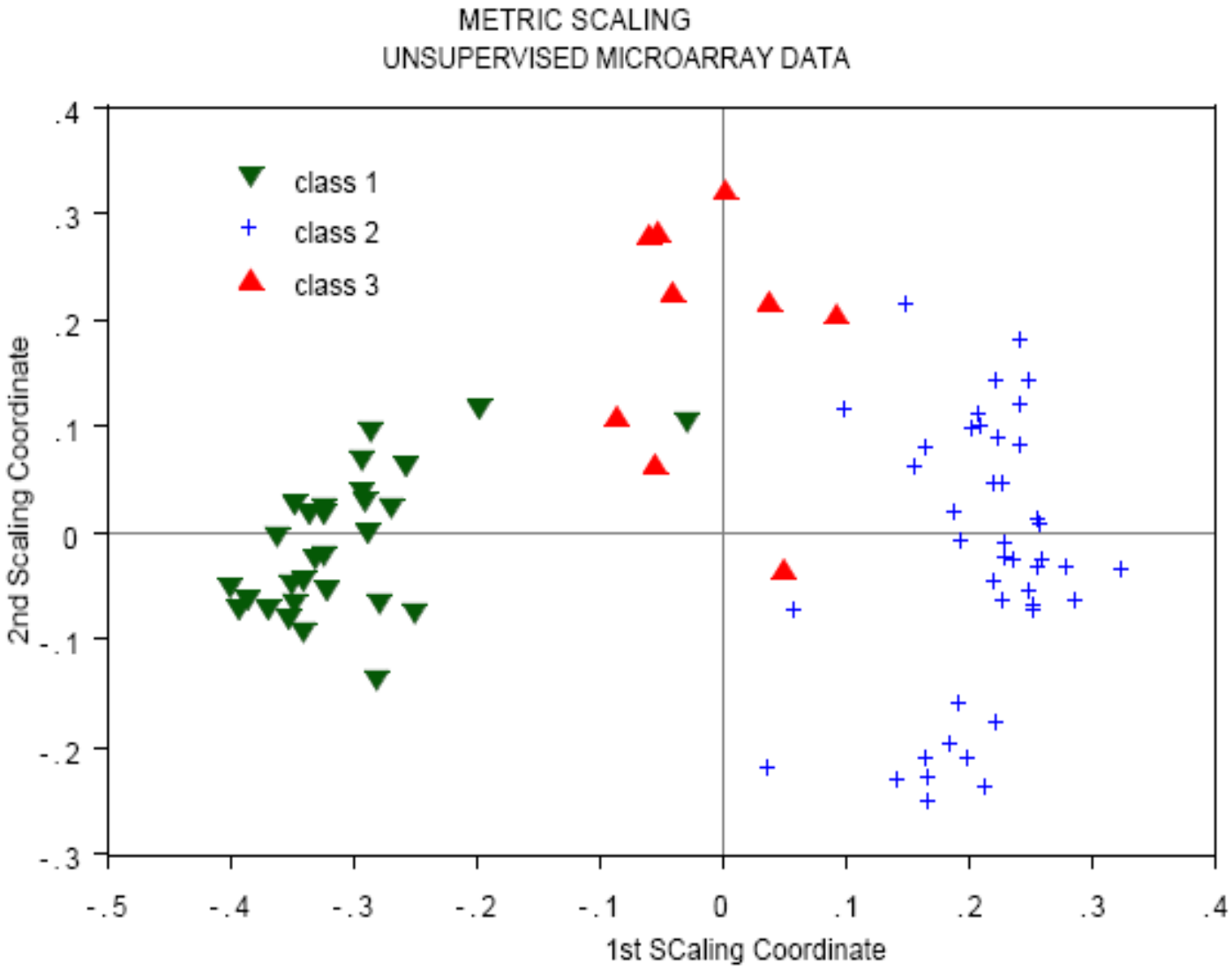


Illustration – Breast Cancer data

- 699 cases, 9 variables, two classes

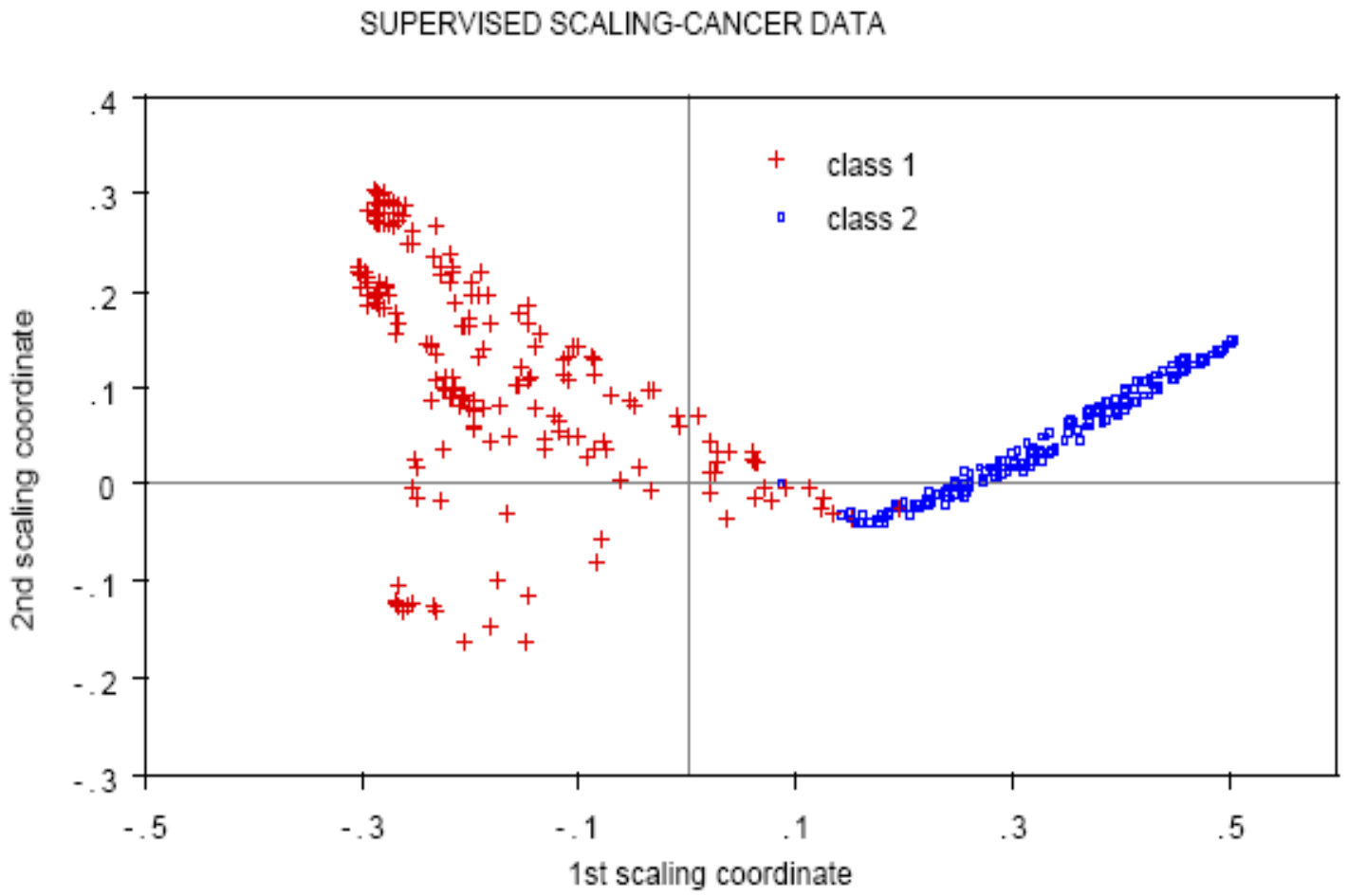


Illustration – Spectra data

The first 468 spectral intensities in the spectra of 764 compounds. Courtesy of Merck.

The challenge: find small cohesive groups of outlying cases.

Excellent separation: error rate of 0.5%.

No outliers were evident in the outlier plots.

However, the scaling picture…

Illustration – Spectra data

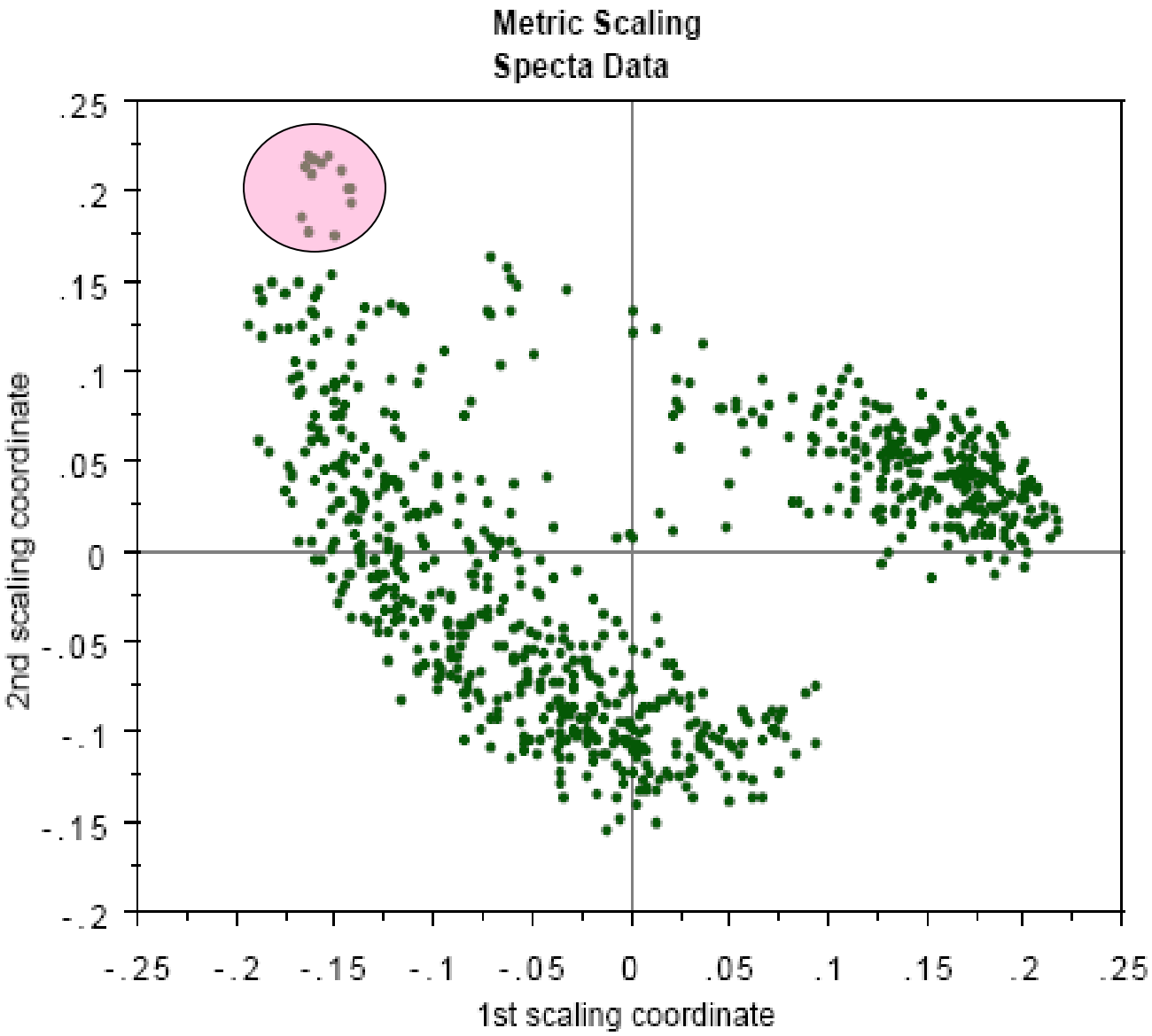
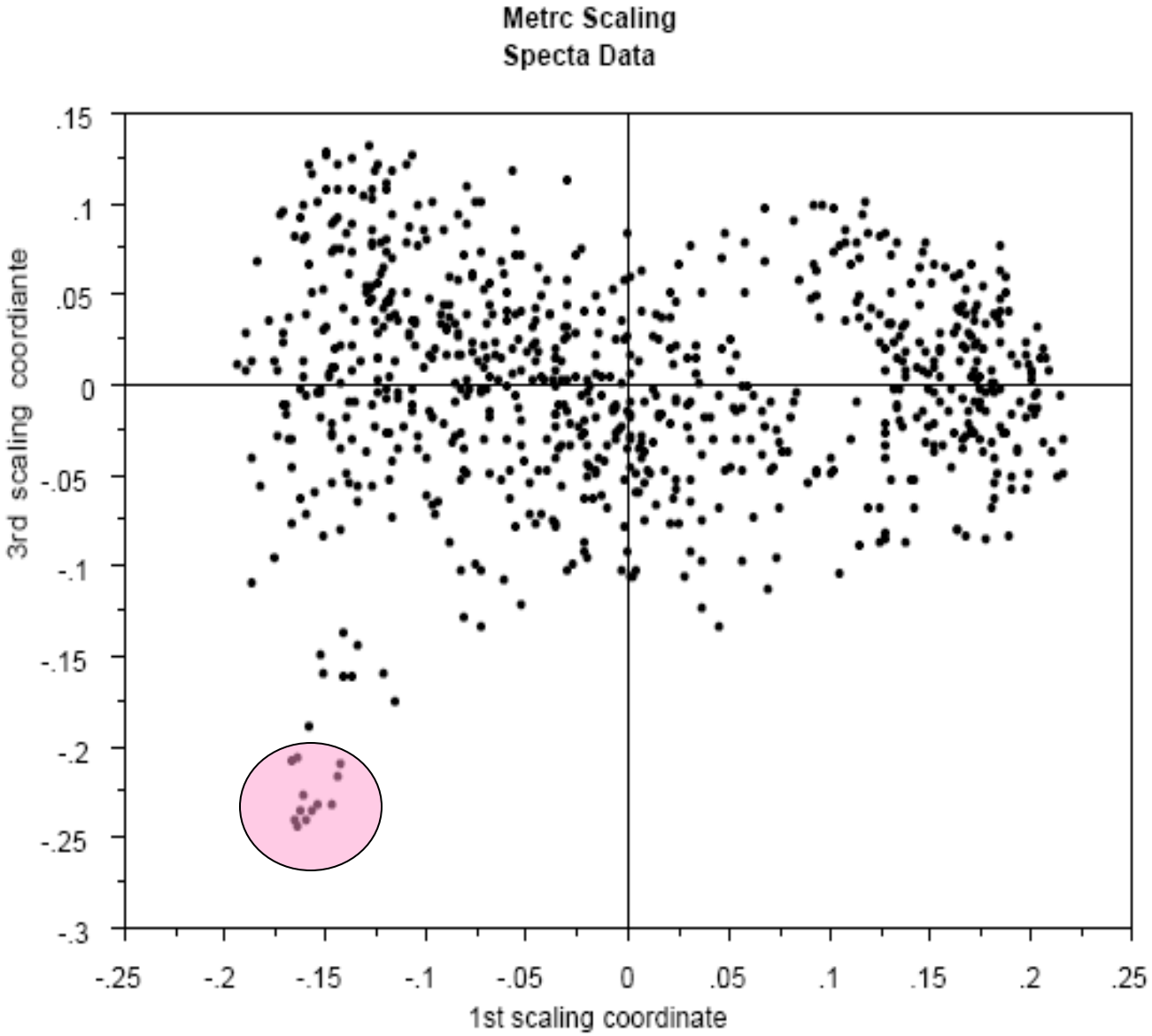


Illustration – Spectra data



Unsupervised learning ≠ clustering

We are trying to discover structure, which may or may not include what we usually think of as “clusters”.

Random forests may allow us to discover structure and may allow us to discover “clusters”.

Current and Future Work

- Proximities
- Detecting interactions
- Regression and Survival Analysis
- Visualization – regression

Visualization

Visualizing proximities

- at-a-glance information about which classes are close, which classes differ
- find clusters within classes
- find easy/hard/unusual cases

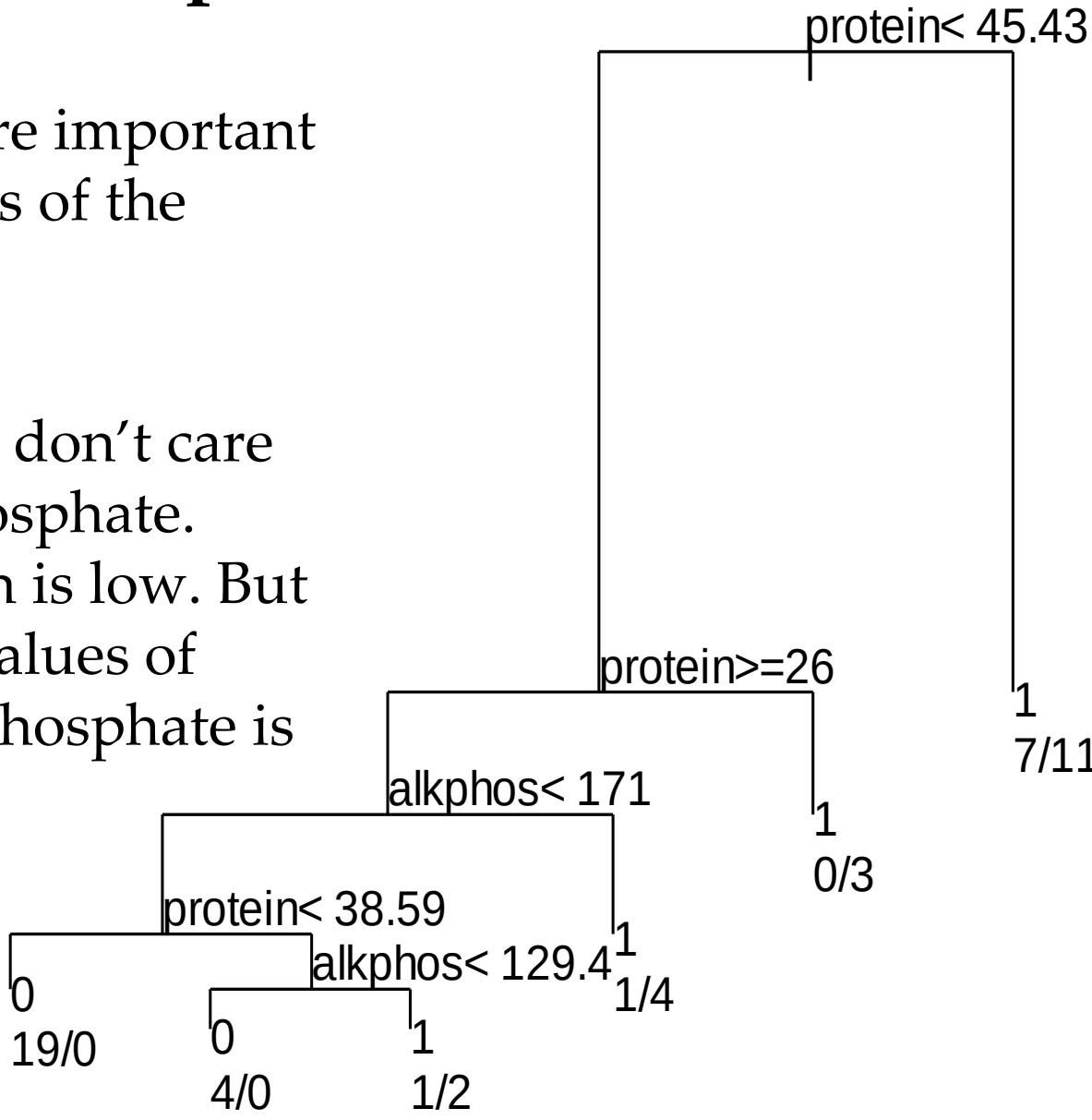
With a good tool we can also

- identify characteristics of unusual points
- see which variables are locally important
- see how clusters or unusual points differ

LOCAL Variable Importance

Different variables are important in different regions of the data.

If protein is high, we don't care about alkaline phosphate. Similarly if protein is low. But for intermediate values of protein, alkaline phosphate is important.



Estimating Local Variable Importance

For each tree, look at the out-of-bag data:

- randomly permute the values of variable *j*, holding the other variables fixed.
- pass these permuted data down the tree, save the classes.

Importance for **case *i*** and variable *j* is

$$\left(\begin{array}{l}\text{error rate } \textbf{for case } \boldsymbol{i} \\ \text{when variable } j \text{ is} \\ \text{permuted}\end{array}\right) - \left(\begin{array}{l}\text{error rate for the} \\ \text{original data}\end{array}\right)$$

where the first error rate is taken over all trees for which case *i* is out-of-bag.

Variable importance for a single class 2 case

TREE	No permutation	Permute variable 1	…	Permute variable m
1	2	2	…	1
3	2	2	…	2
4	1	1	…	1
9	2	2	…	1
…	…	…	…	…
992	2	2	…	2
% Error	10%	11%	…	35%

Case Studies

Microarrays

Ramón Díaz-Uriarte, Sara Alvarez de Andrés
Bioinformatics Unit, Spanish National Cancer Center
March, 2005 http://ligarto.org/rdiaz

Compared

- SVM, linear kernel
- KNN/crossvalidation (Dudoit et al. JASA 2002)
- DLDA
- Shrunken Centroids (Tibshirani et al. PNAS 2002)
- Random forests

"Given its performance, random forest and variable selection using random forest should probably become part of the standard tool-box of methods for the analysis of microarray data."

Microarray Error Rates

Data	*M*	*N*	*Class*	*SVM*	*KNN*	*DLDA*	*SC*	*RF*
Leukemia	3051	38	2	.014	.029	.020	.025	.051
Breast 2	4869	78	2	.325	.337	.331	.324	.342
Breast 3	4869	96	3	.380	.449	.370	.396	.351
NCI60	5244	61	8	.256	.317	.286	.256	.252
Adenocar	9868	76	2	.203	.174	.194	.177	.125
Brain	5597	42	5	.138	.174	.183	.163	.154
Colon	2000	62	2	.147	.152	.137	.123	.127
Lymphoma	4026	62	3	.010	.008	.021	.028	.009
Prostate	6033	102	2	.064	.100	.149	.088	.077
Srbct	2308	63	4	.017	.023	.011	.012	.021

Microarray Error Rates

Data	*SVM*	*KNN*	*DLDA*	*SC*	*RF*
Leukemia	.014	.029	.020	.025	.051
Breast 2	.325	.337	.331	.324	.342
Breast 3	.380	.449	.370	.396	.351
NCI60	.256	.317	.286	.256	.252
Adenocar	.203	.174	.194	.177	.125
Brain	.138	.174	.183	.163	.154
Colon	.147	.152	.137	.123	.127
Lymphoma	.010	.008	.021	.028	.009
Prostate	.064	.100	.149	.088	.077
Srbct	.017	.023	.011	.012	.021
Mean	**.155**	**.176**	**.170**	**.159**	**.151**

Case study - Brain Cancer Microarrays

Pomeroy et al. Nature, 2002.

Dettling and Bühlmann, Genome Biology, 2002.

42 cases, 5,597 genes, 5 tumor types:

- 10 medulloblastomas BLUE
- 10 malignant gliomas PALE BLUE
- 10 atypical teratoid/rhabdoid tumors (AT/RTs) GREEN
- 4 human cerebella ORANGE
- 8 PNETs RED

Case Study - Autism

Data courtesy of J.D.Odell and R. Torres, USU

154 subjects (308 chromosomes)

7 variables, all categorical (up to 30 categories)

2 classes:

- Normal, blue (69 subjects)
- Autistic, red (85 subjects)

Case Study – Invasive Plants

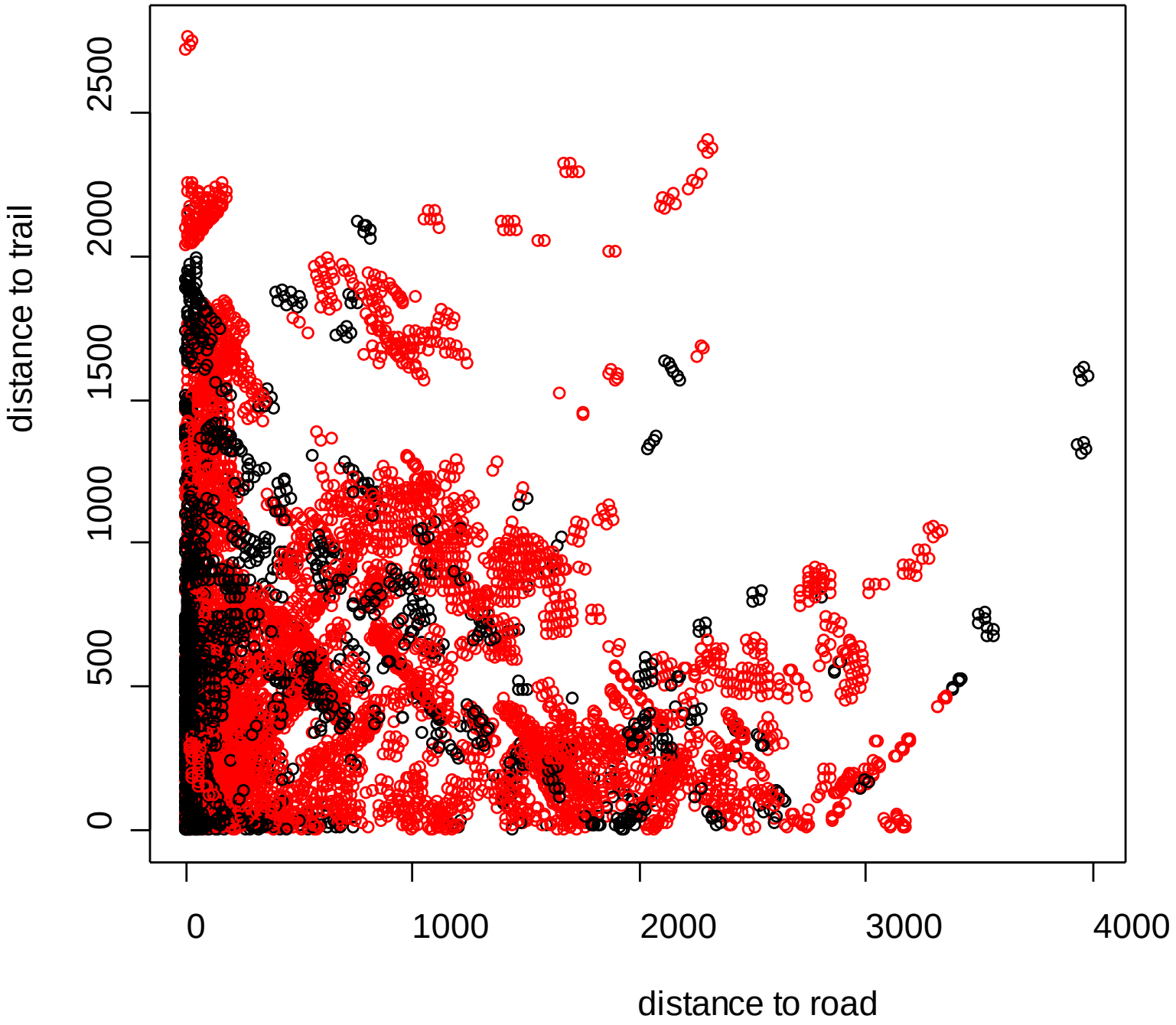
Data courtesy of Richard Cutler, Tom Edwards

8251 cases

30 variables

2 classes:

- Absent, blue (2204 cases)
- Present, red (6047 cases)


distance to trail
0
500
1000
1500
2000
2500
0
1000
2000
3000
4000
distance to road

Initial run, m=5, equal weights

Error rate = 5%

Out-of-bag confusion matrix

	Absent	**Present**
Called absent	1974	183
Called present	230	5864
Total	2204	6047
Error rate	**10%**	**3%**

Second run, m=5, weights .75 and .25

Error rate = 6.5%

Out-of-bag confusion matrix

	Absent	**Present**
Called absent	2088	420
Called present	116	5627
Total	2204	6047
Error rate	**5.3%**	**7.0%**

Third run, m=5, weights .7 and .3

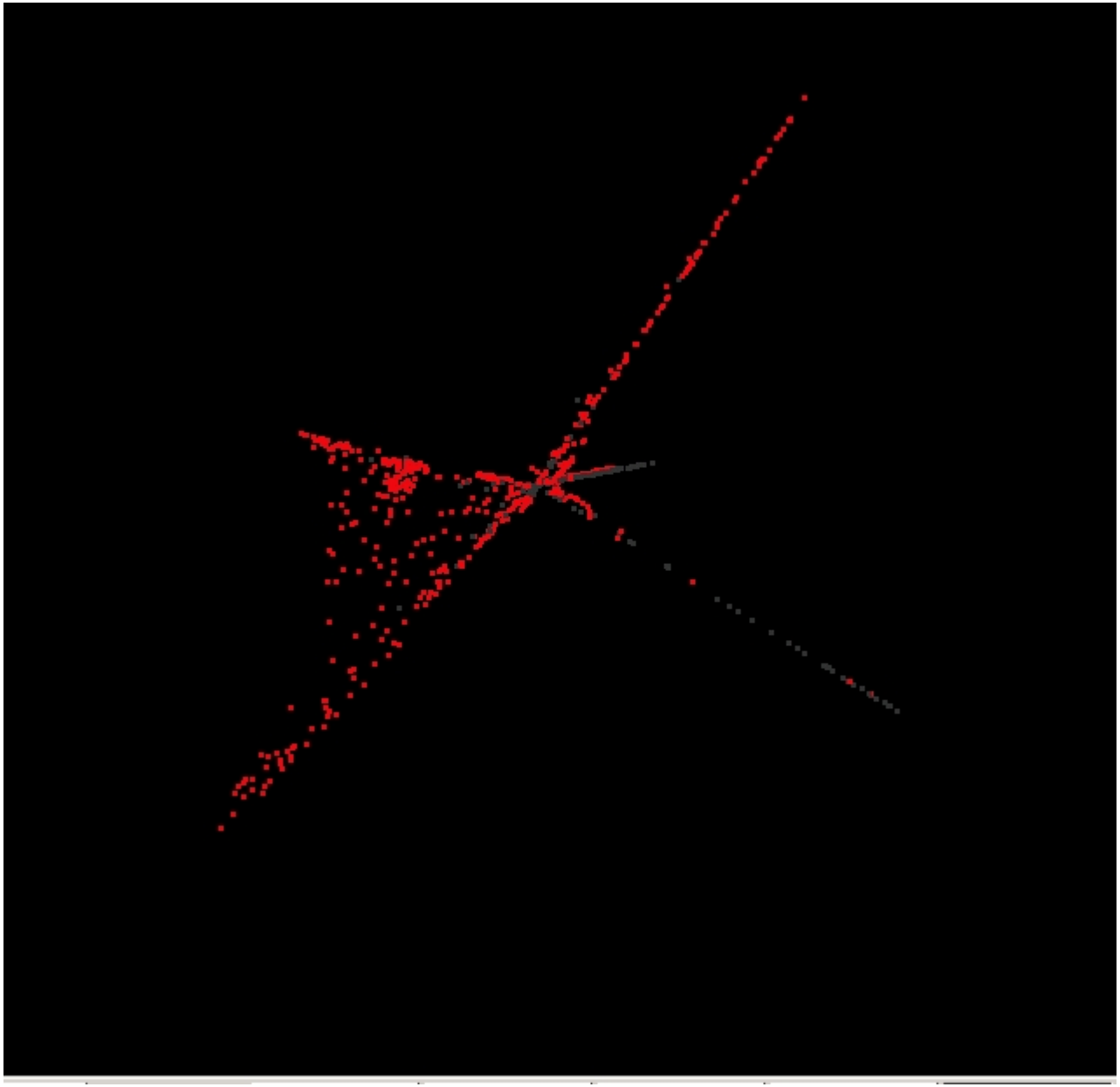
Error rate = 5.9 confusion matrix

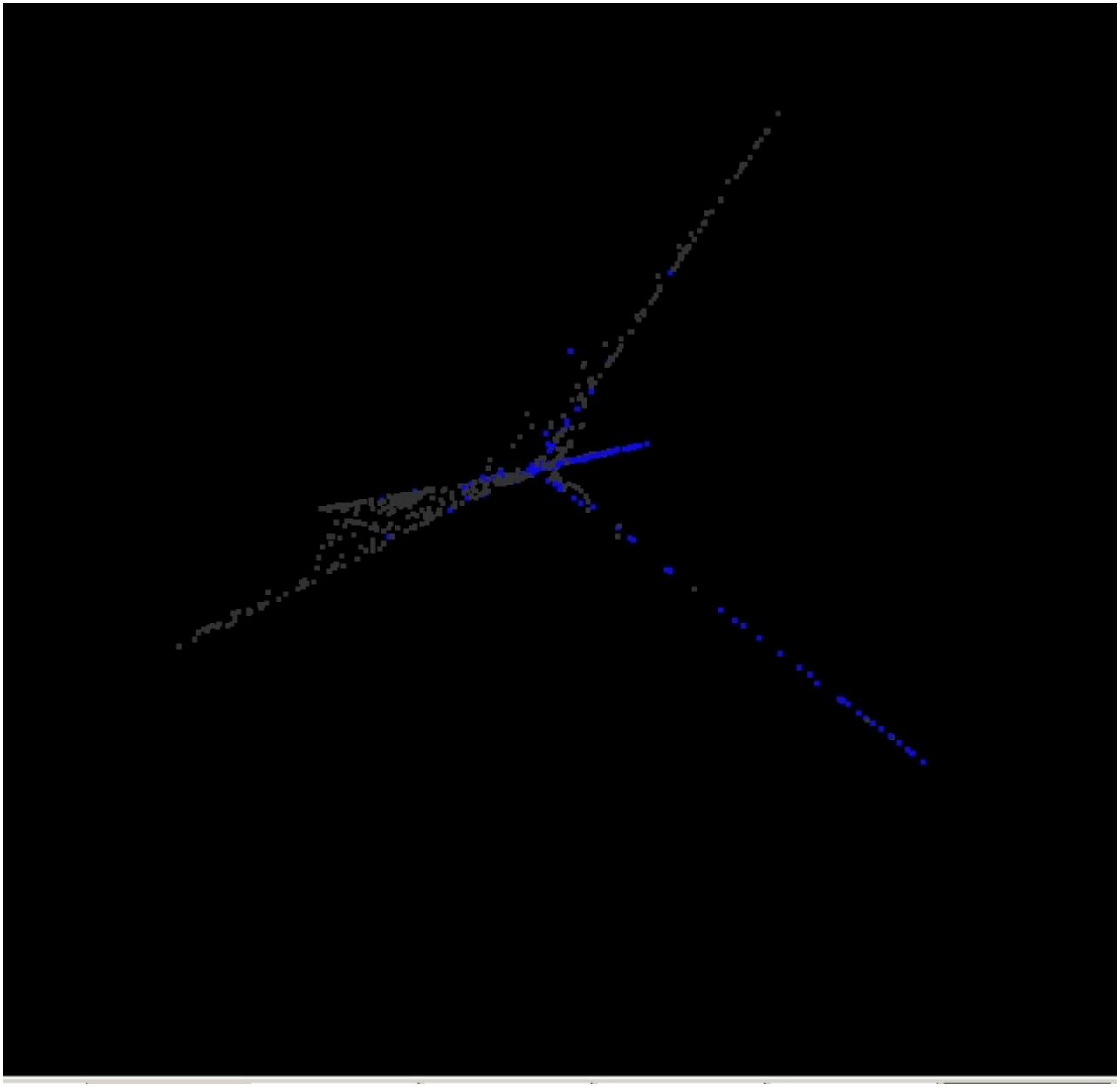
	Absent	**Present**
Called absent	2066	347
Called present	138	5700
Total	2204	6047
Error rate	**6.3%**	**5.7%**

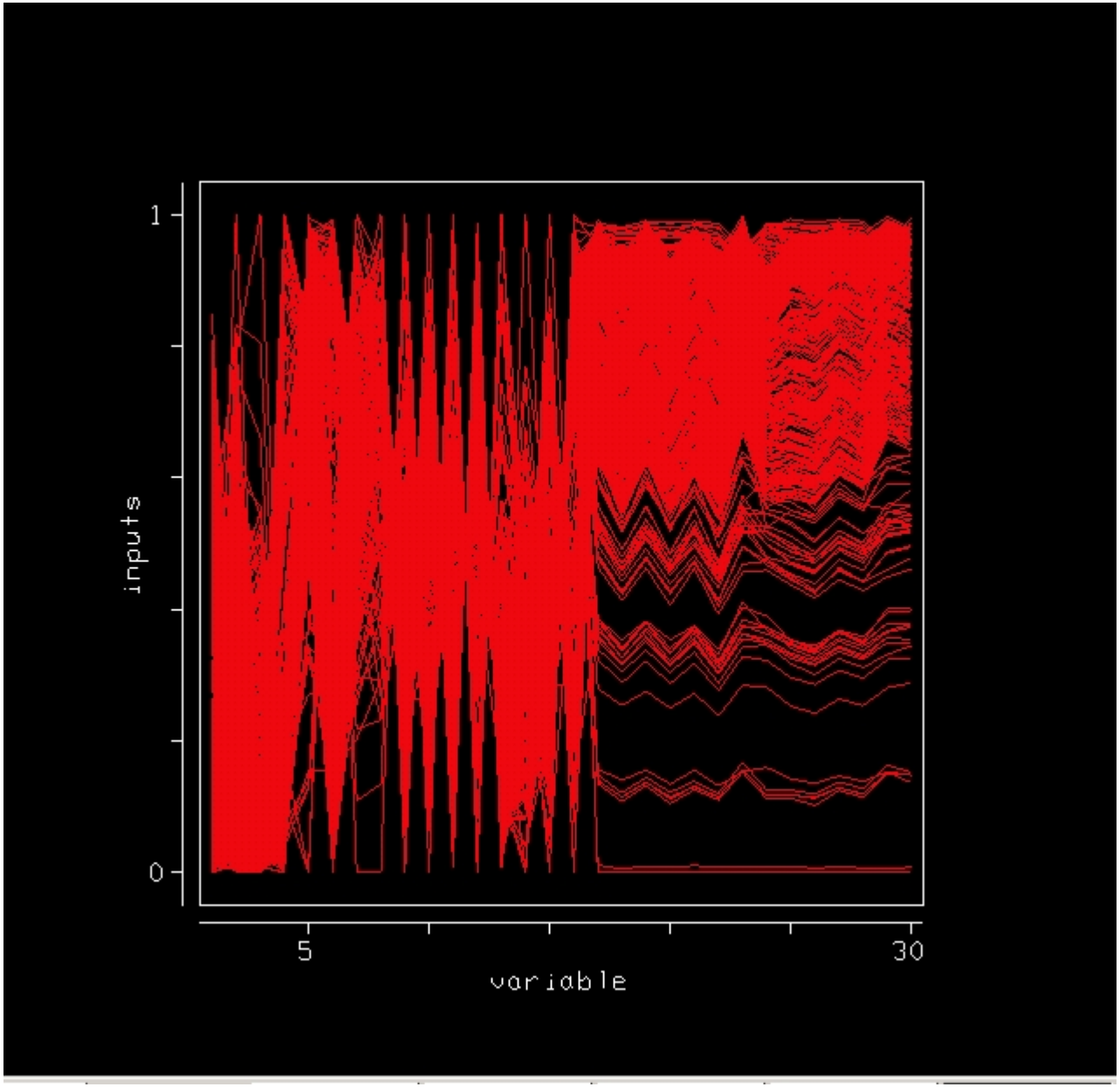
Last run, m=5, weights .72 and .28

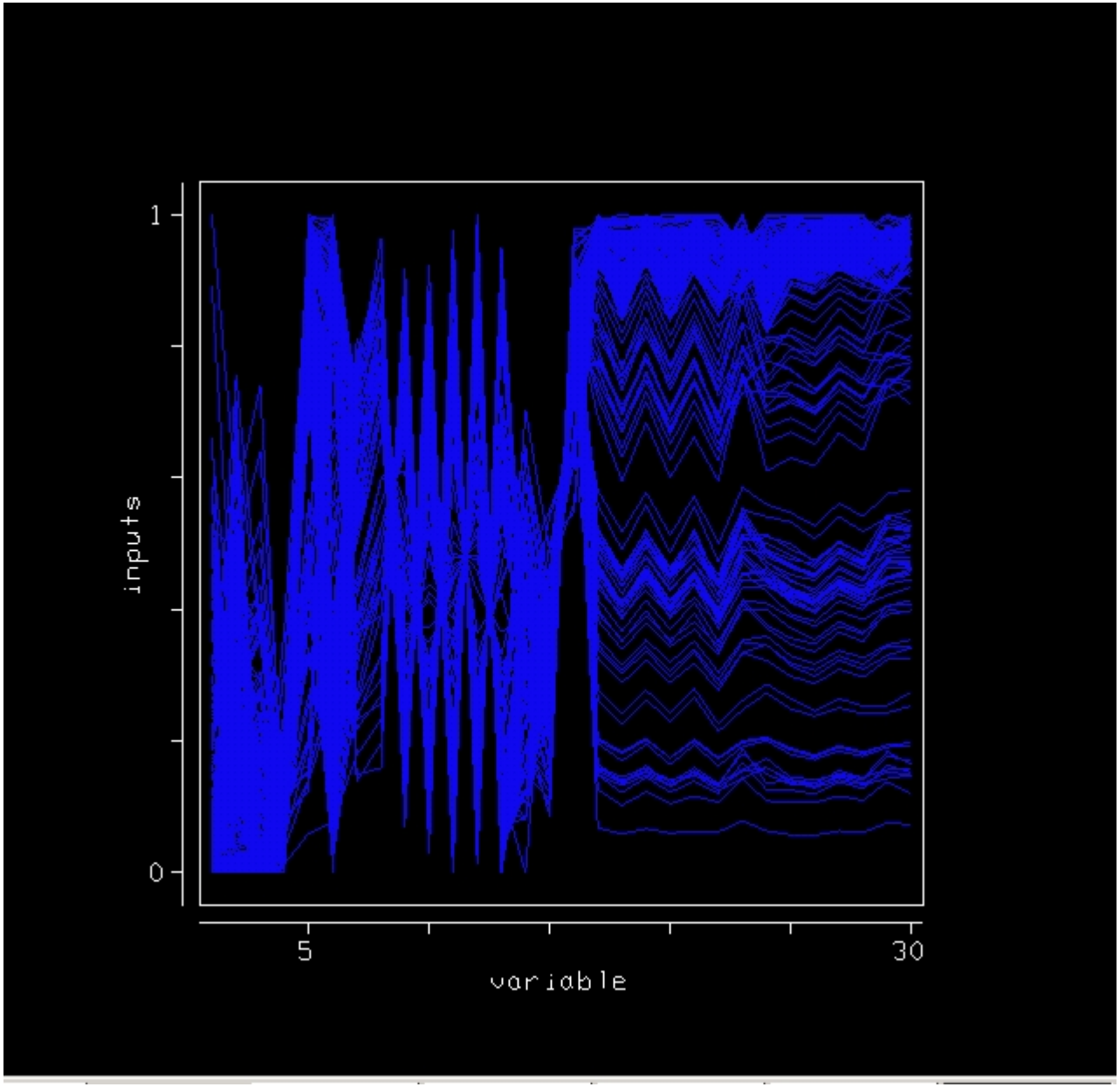
Error rate = 6.17 confusion matrix

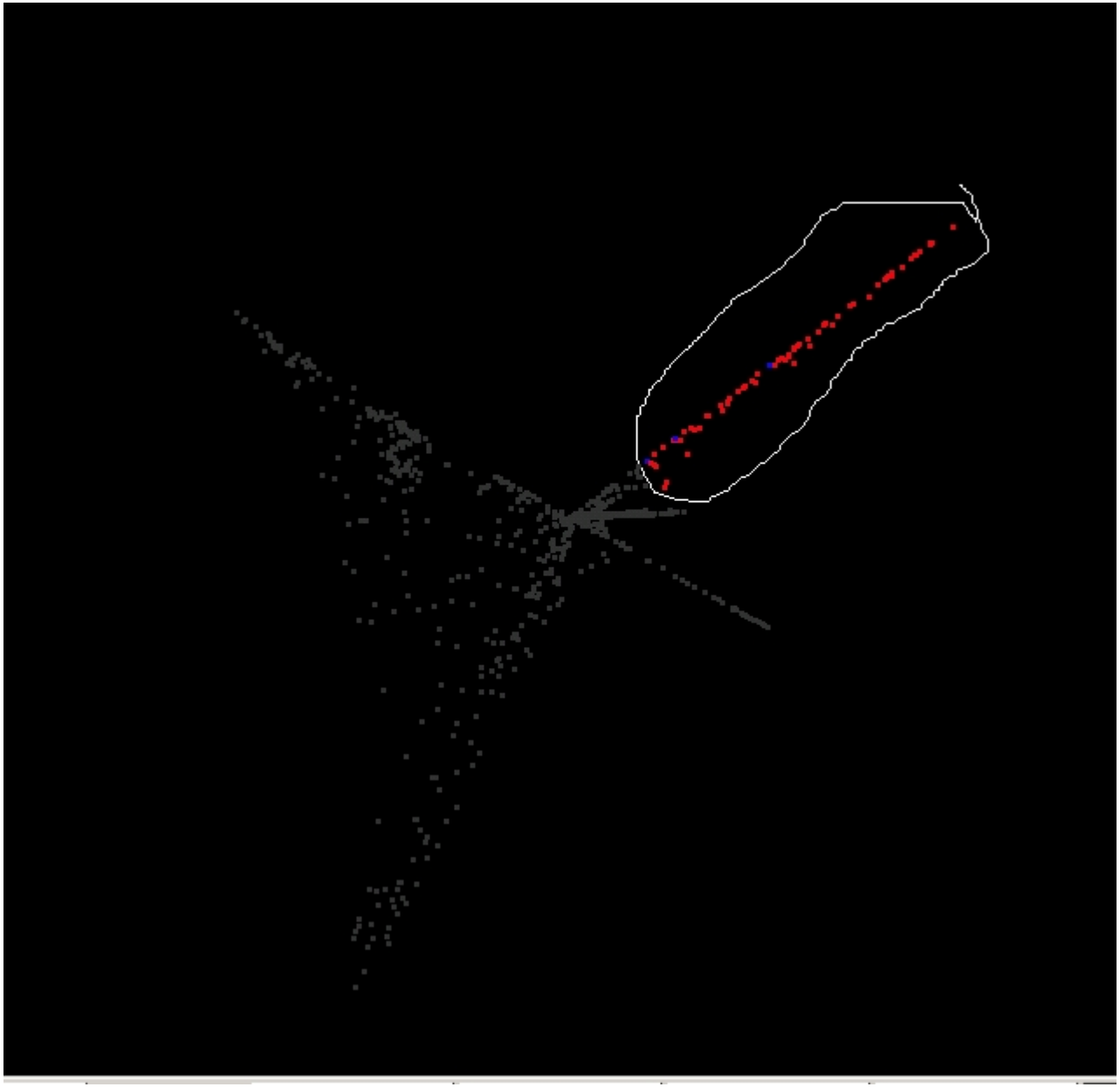
	Absent	**Present**
Called absent	2072	377
Called present	132	5670
Total	2204	6047
Error rate	**6.0%**	**6.2%**

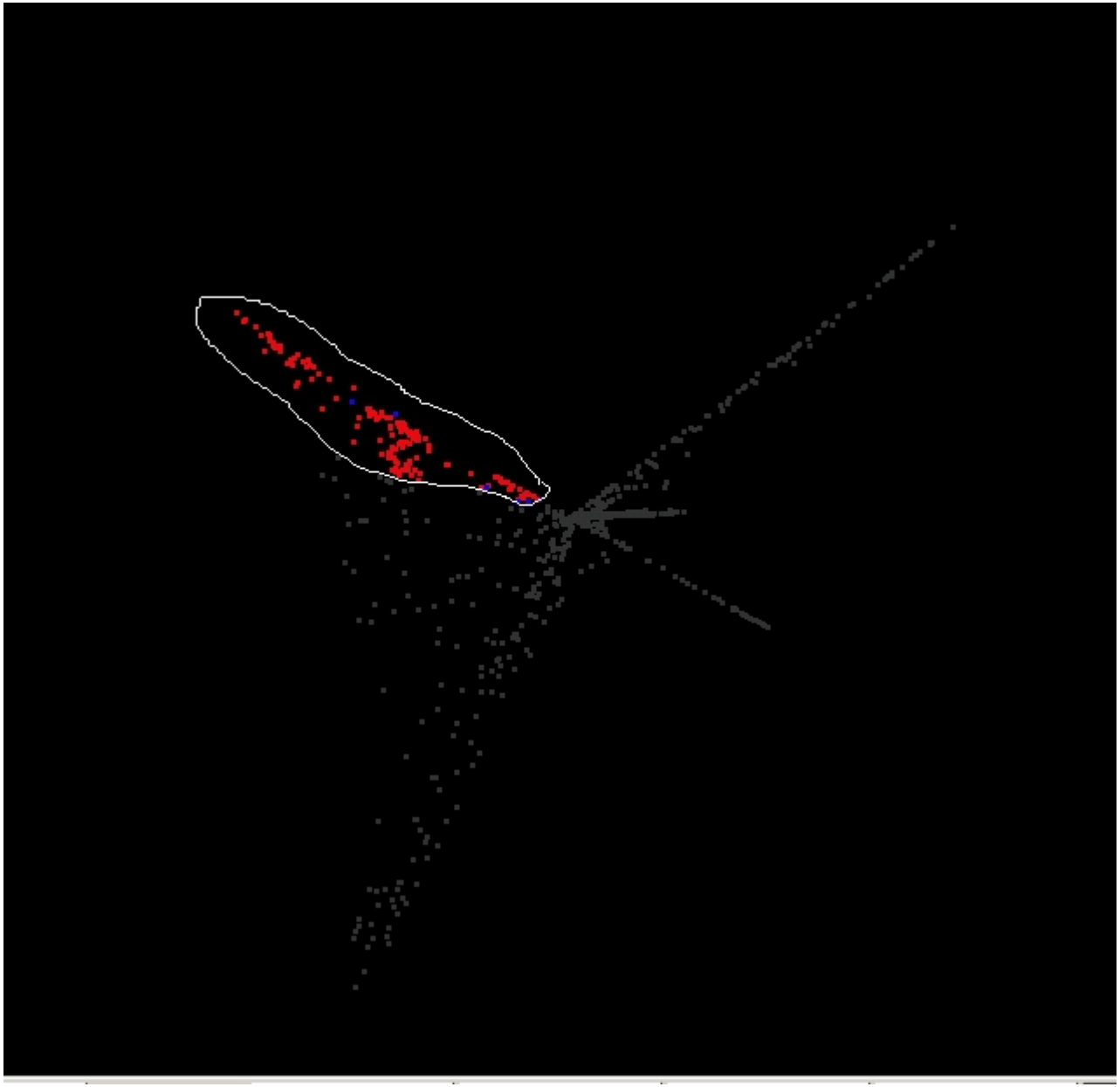


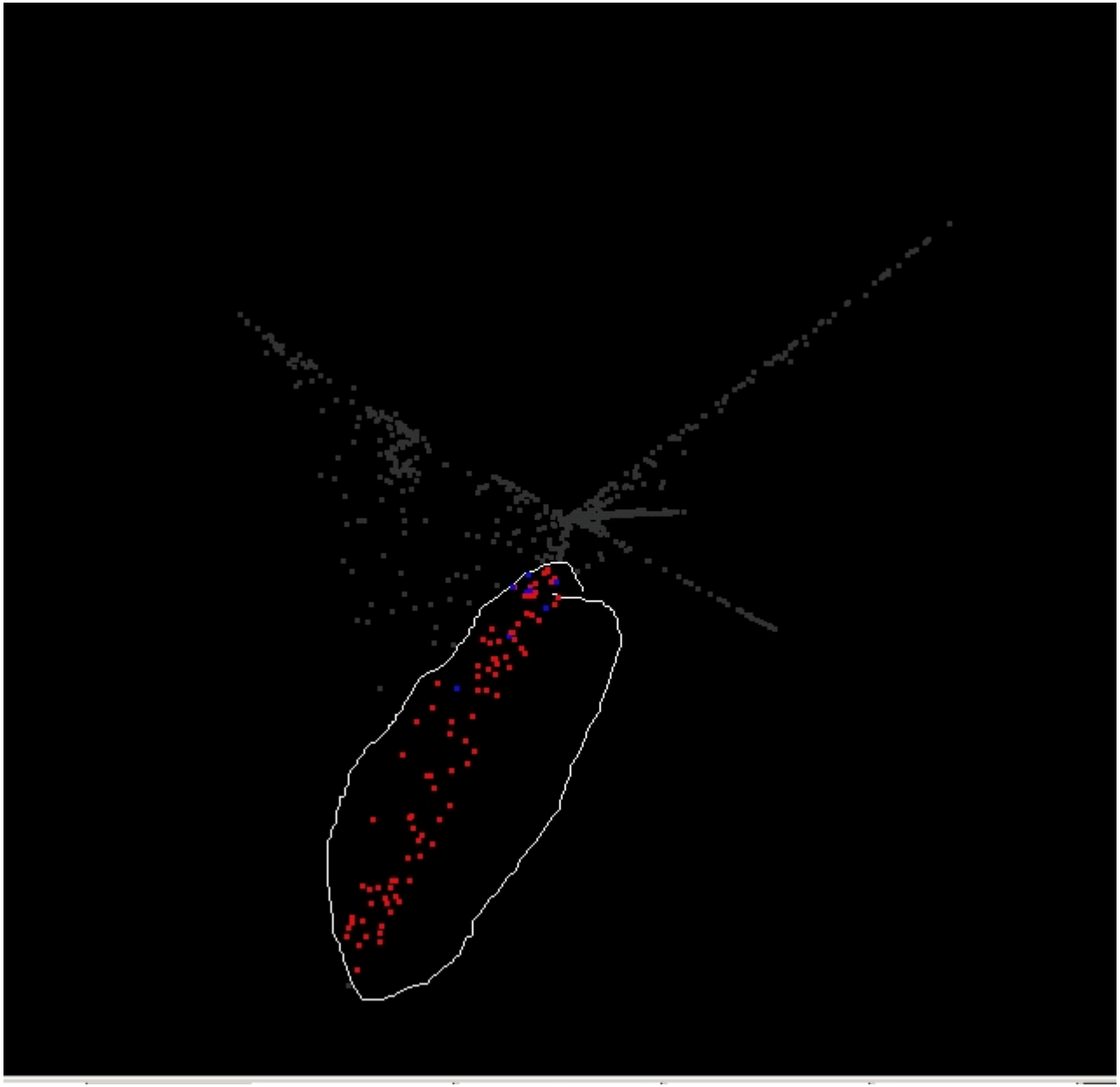


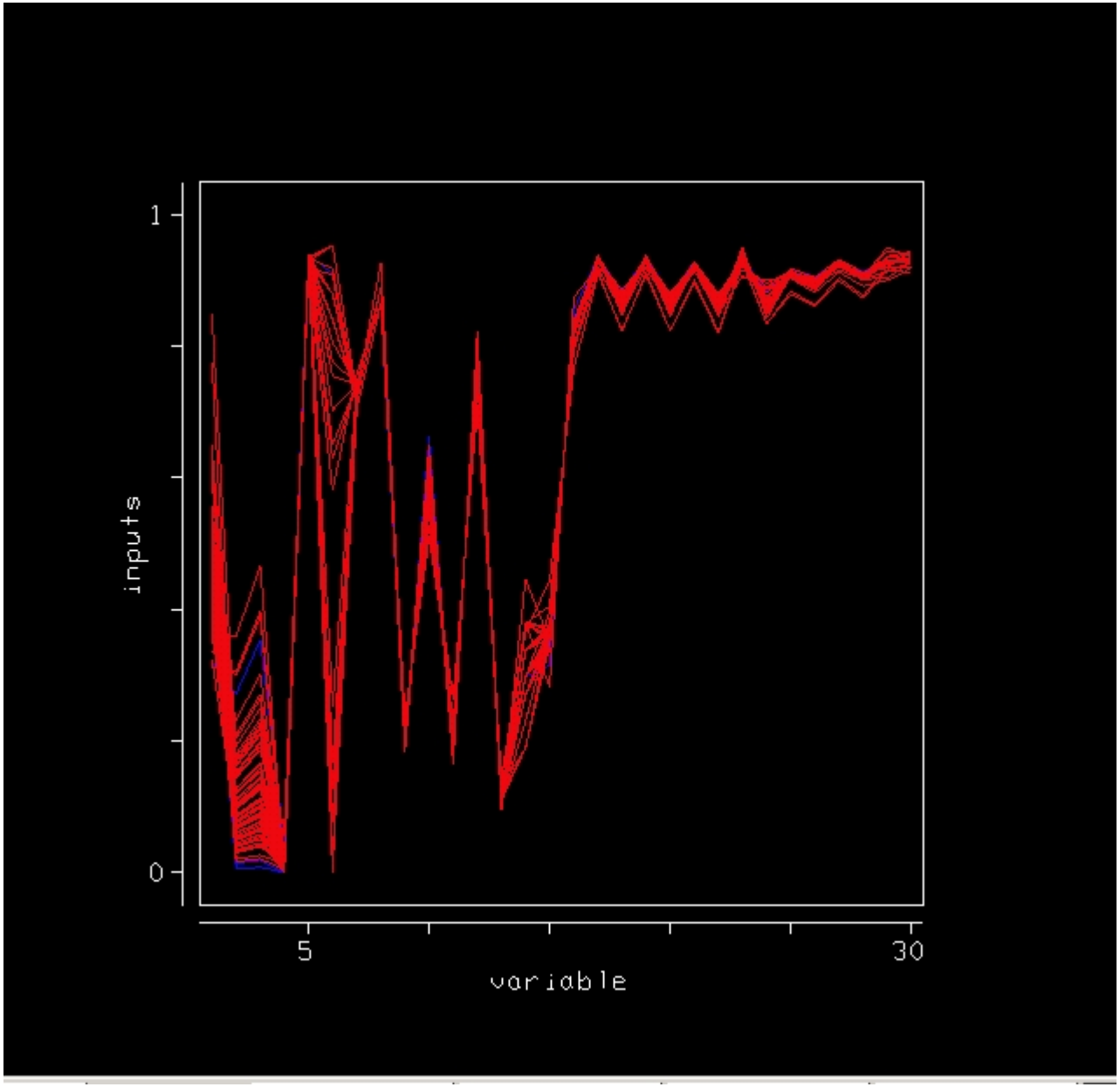

1
0
inputs
5
30
variable

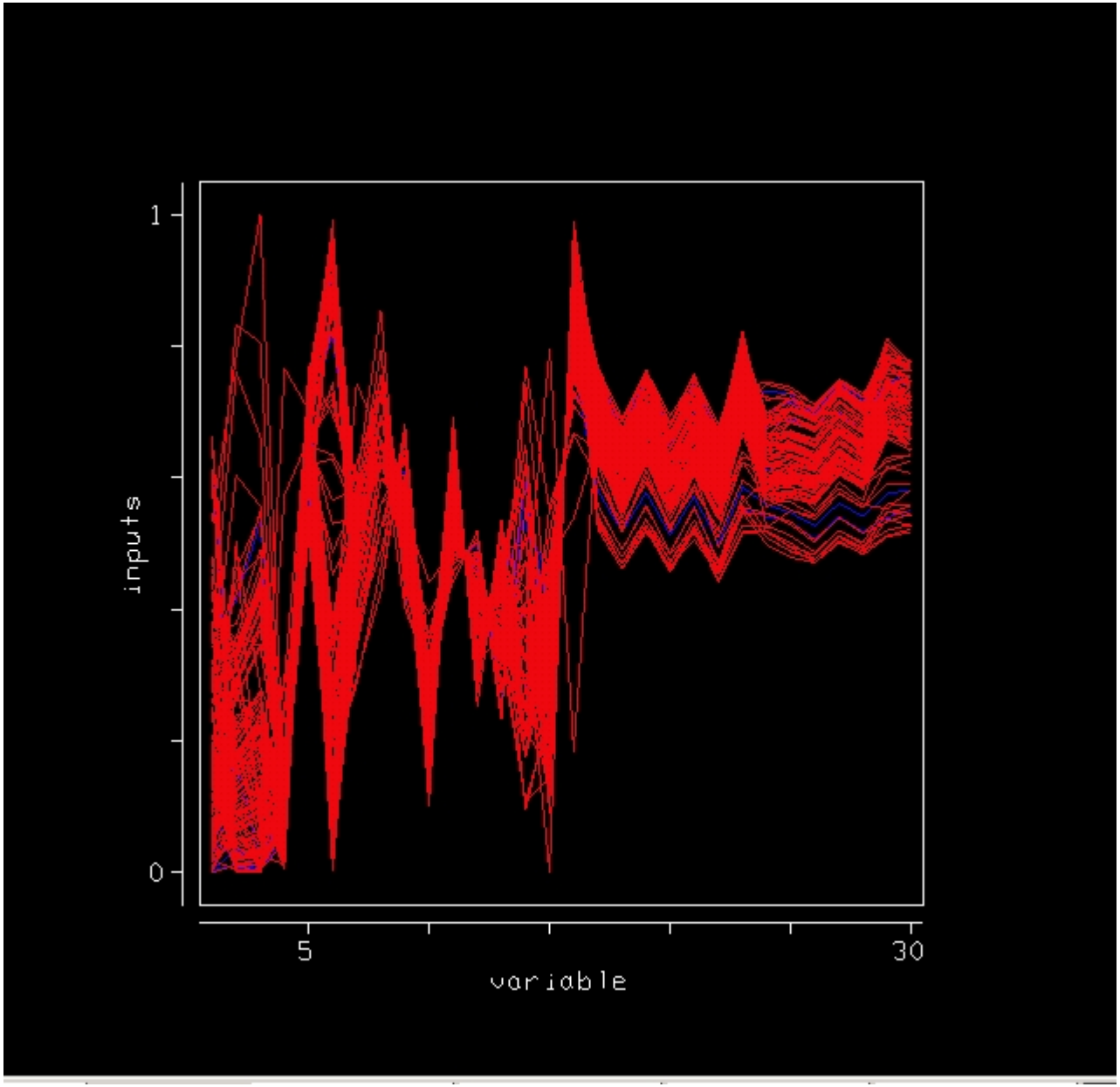

1
0
inputs
5
30
variable

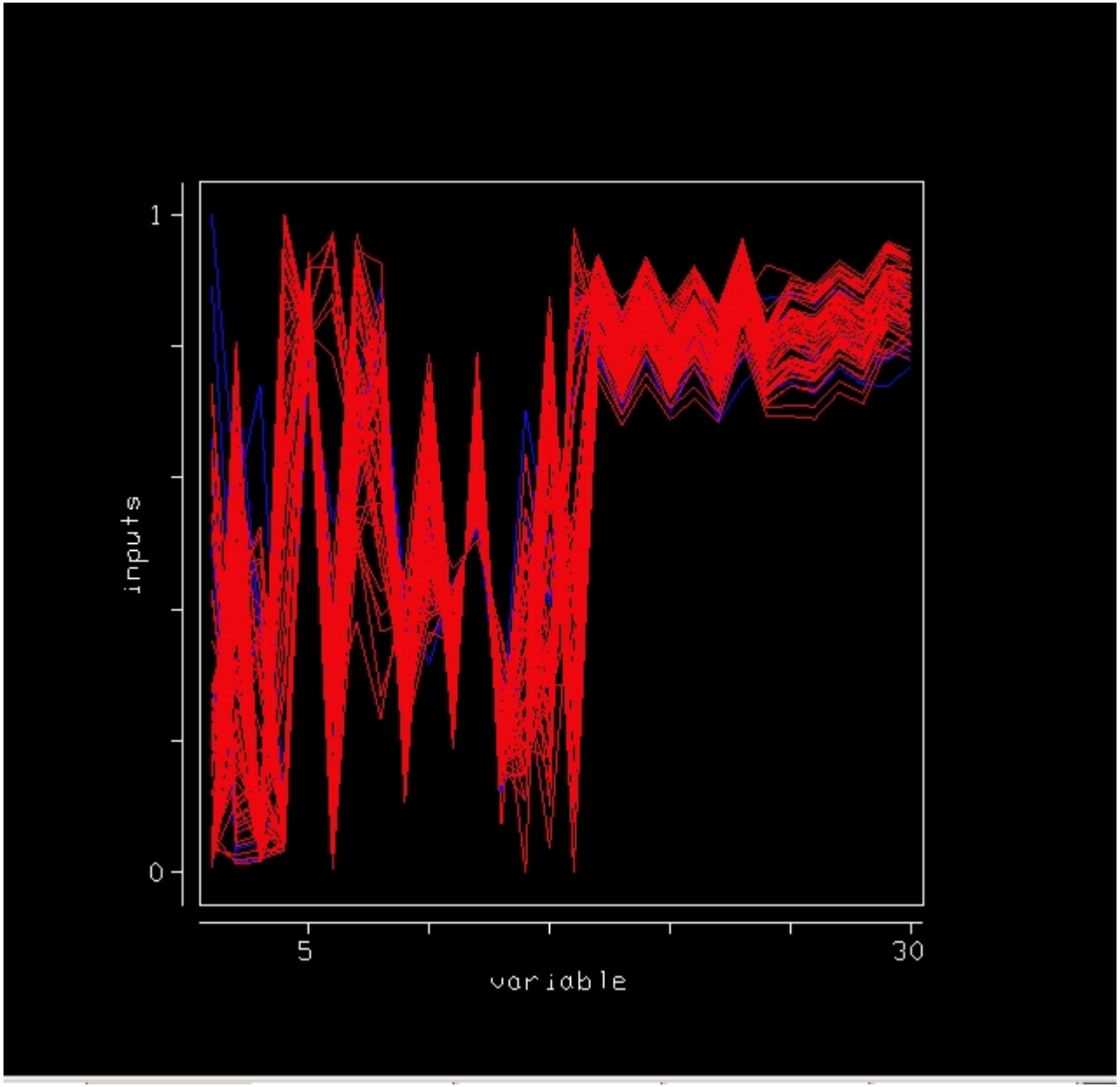


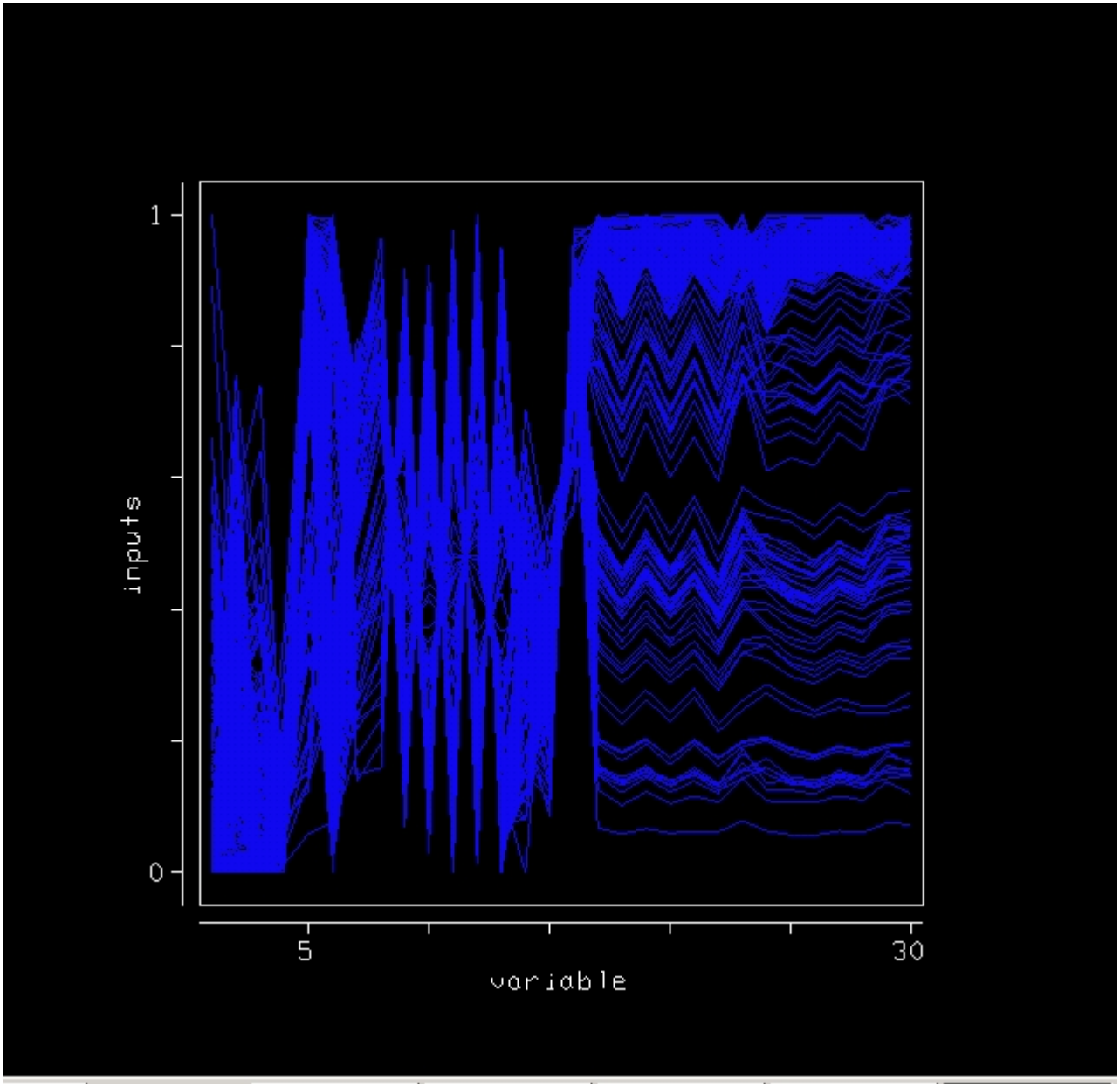





1
0
inputs
5
30
variable


1
0
inputs
5
30
variable


1
0
inputs
5
30
variable


1
0
inputs
5
30
variable