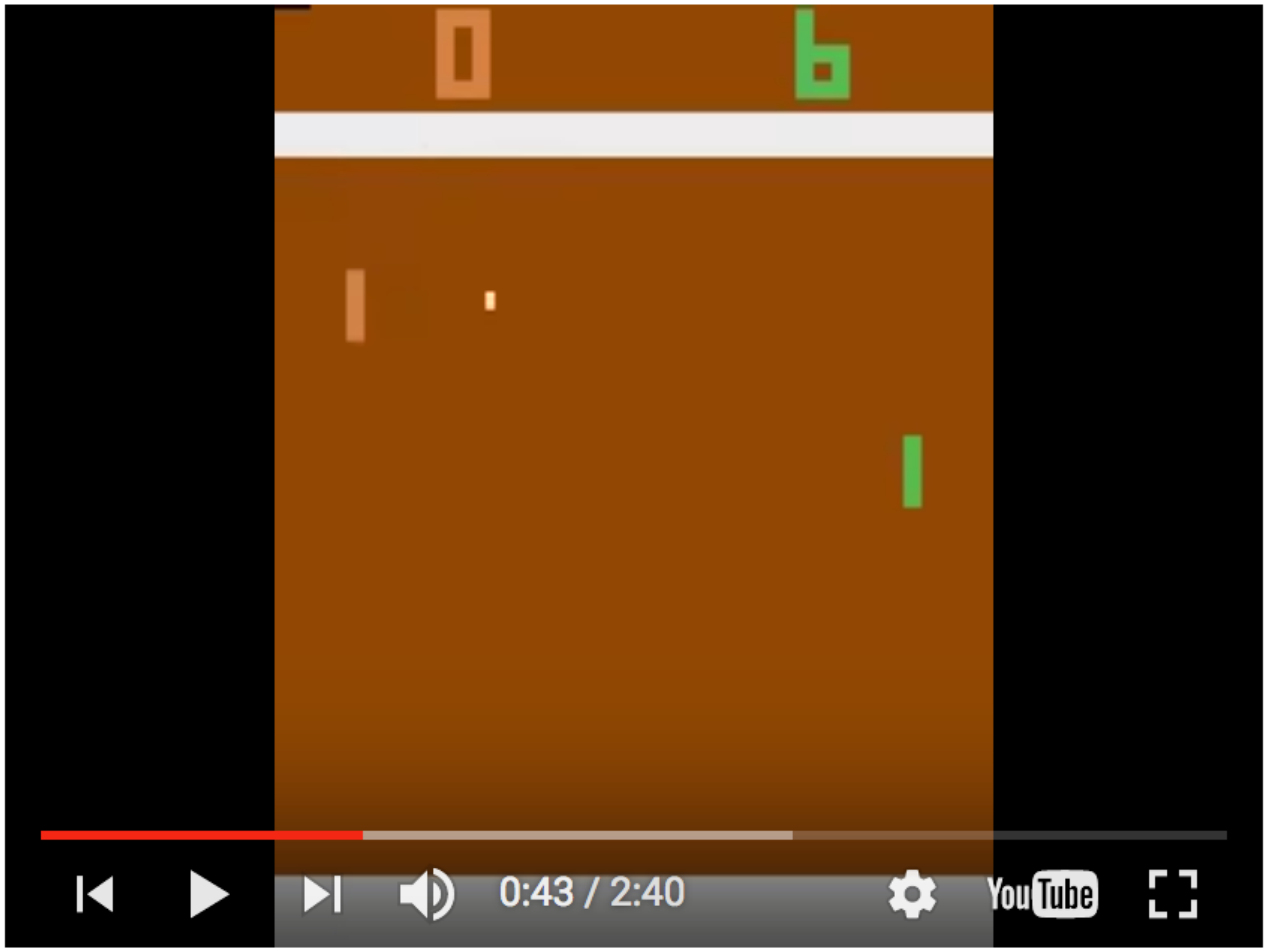
Data management and Machine Learning

BIOINF 525

Session 3, Lecture 4

4/11/2017

Classic Computer player



Neural network

http://karpathy.github.io/2016/05/31/rl/

NATURE | LETTER

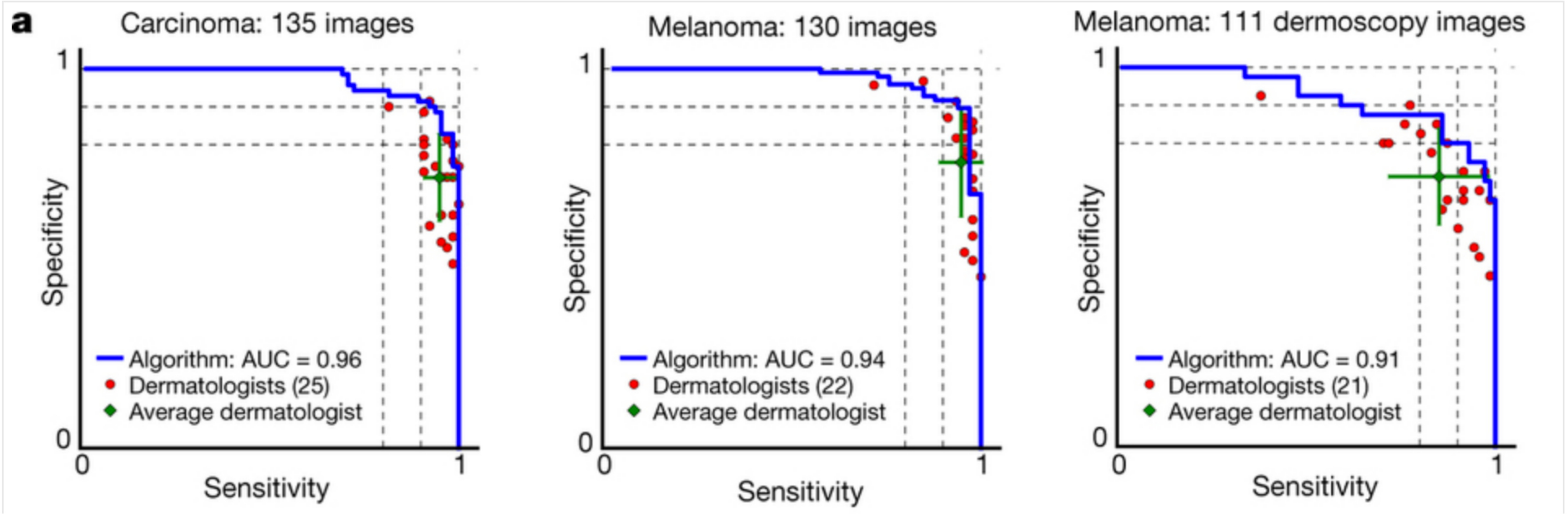
日本語要約

Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun

Affiliations | Contributions | Corresponding authors

Nature **542**, 115–118 (02 February 2017) | doi:10.1038/nature21056



Outline

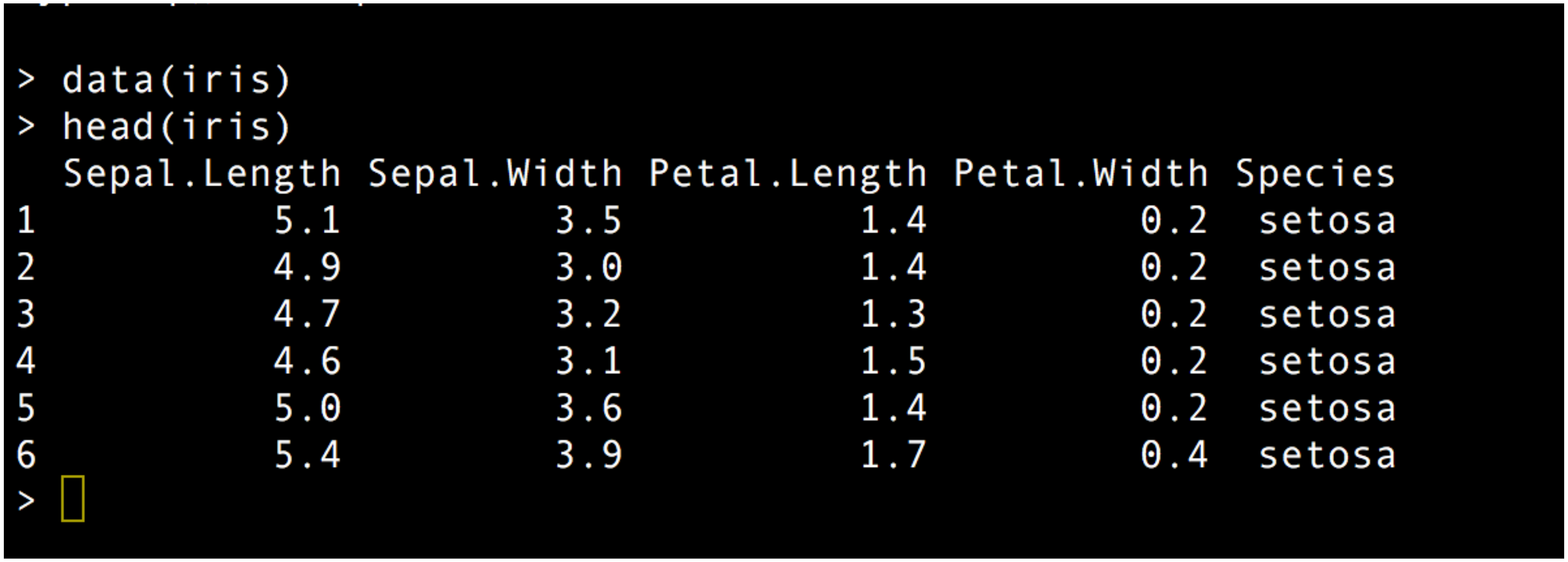
- Data import and management in R
- Overview of machine learning
- Common machine learning methods
- Applications of machine learning in biology

Outline

- **Data import and management in R**
- Overview of machine learning
- Common machine learning methods
- Applications of machine learning in biology

Data import and management in R

Data frames are the fundamental data structure of R



```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> 
```

Data import and management in R

Data frames can be generated by hand:

```
qpcr.dat = data.frame(sample.id = samples, gene.id = targets, ddct = ddct.vals)
```

Data frames can be imported from a csv file or text file:

```
data.1 = read.csv("my_data.csv")
data.2 = read.table("my_data_2.txt”, header=TRUE, sep=";")
```

Data frames can be read using functions of specific modules:

```
dds = DESeqDataSetFromMatrix(countData=count.mat,colData=samptab,
design=~media+starvation+media:starvation)
```

Uniform requirements for R input files

- Using a **plain** text editor is essential: gedit, textwrangler, vim, emacs…
- Different platforms have different line ending types; look at the programs dos2unix and mac2unix if you see problems
- Be wary of “smart” quotes, hyphens, etc. – only a limit set of text characters is allowed
- Similar precautions for programming, and for other structured text files like PDBs

Table-like vs. matrix-like data frames

Matrix like

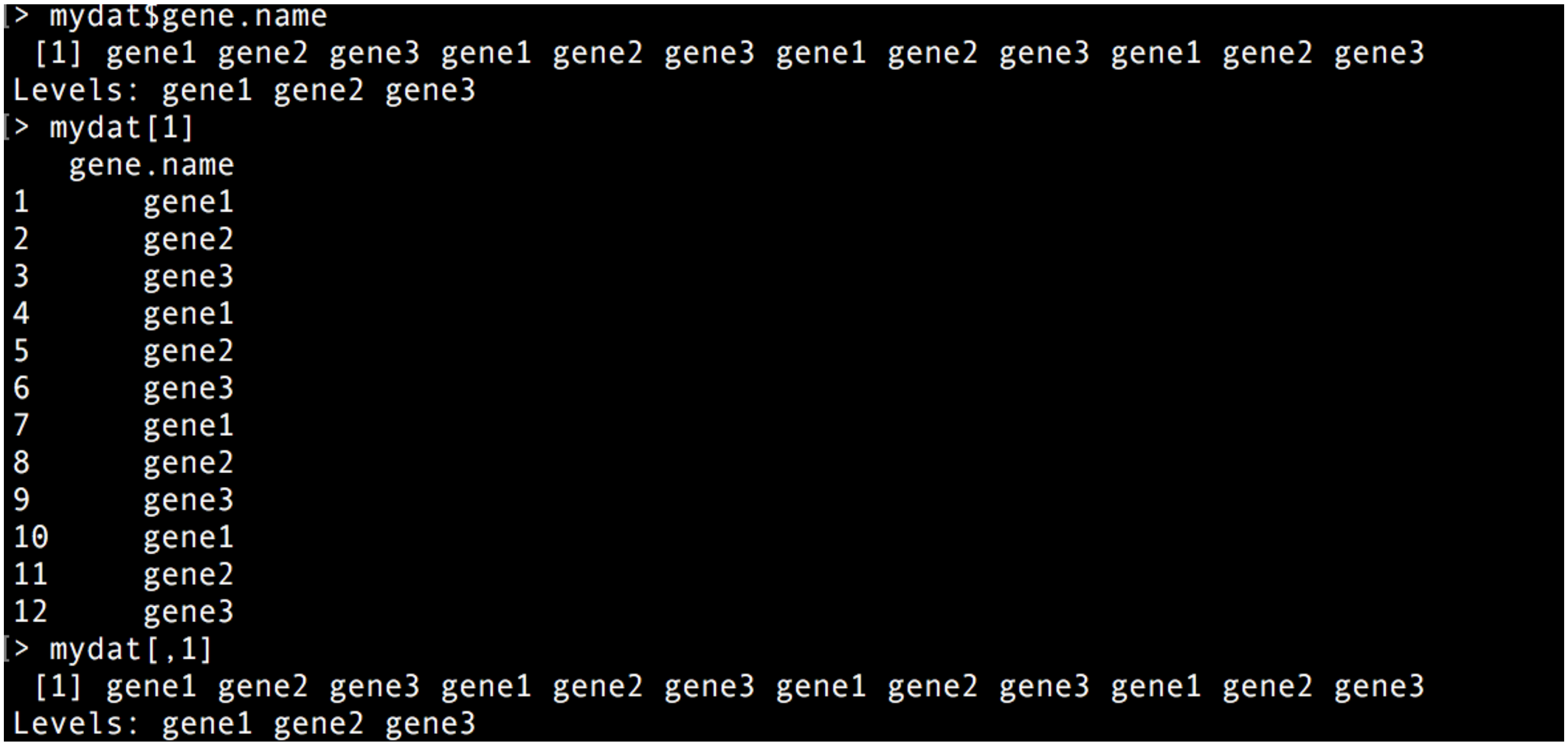
```
gene.name,A1,A2,B1,B2
gene1,4,3,10,8
gene2,4,6,5,3
gene3,2,9,10,0
```

Table like

```
gene.name,condition,value,replicate.id
gene1,A,4,1
gene2,A,4,1
gene3,A,2,1
gene1,A,3,2
gene2,A,6,2
gene3,A,9,2
gene1,B,10,1
gene2,B,5,1
gene3,B,10,1
gene1,B,8,2
gene2,B,3,2
gene3,B,0,2
```

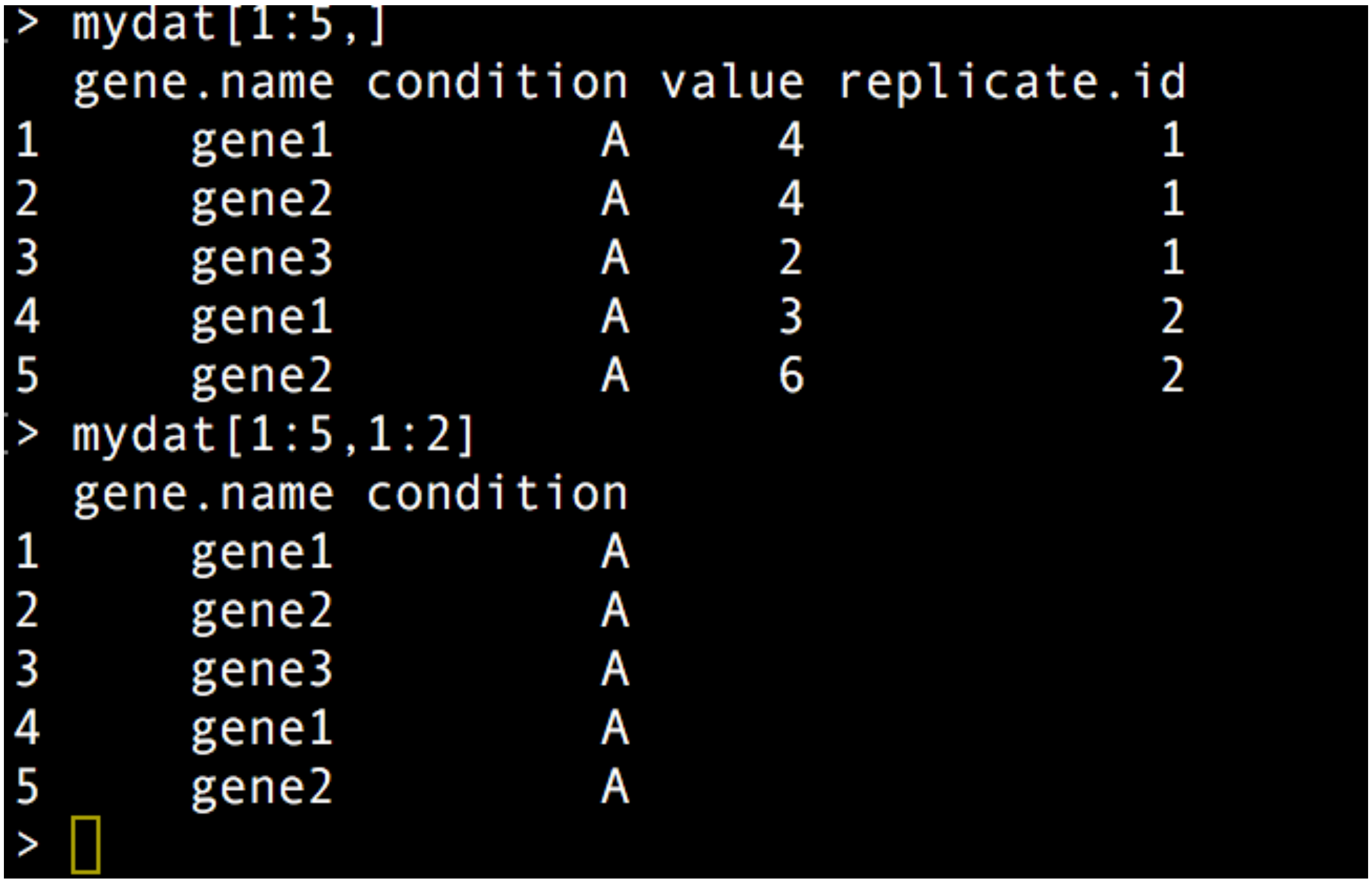
These can be interconverted using the reshape2 R package

Accessing data in a data frame



```
> mydat$gene.name
 [1] gene1 gene2 gene3 gene1 gene2 gene3 gene1 gene2 gene3 gene1 gene2 gene3
Levels: gene1 gene2 gene3
> mydat[1]
   gene.name
1      gene1
2      gene2
3      gene3
4      gene1
5      gene2
6      gene3
7      gene1
8      gene2
9      gene3
10     gene1
11     gene2
12     gene3
> mydat[,1]
 [1] gene1 gene2 gene3 gene1 gene2 gene3 gene1 gene2 gene3 gene1 gene2 gene3
Levels: gene1 gene2 gene3
```

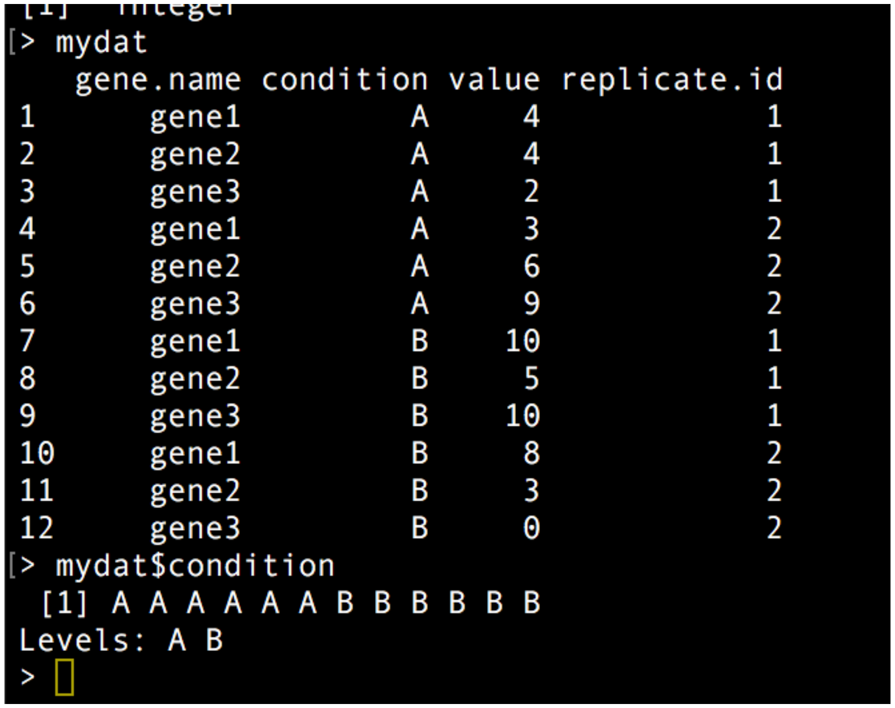
Accessing data in a data frame



```
> mydat[1:5,]
  gene.name condition value replicate.id
1     gene1         A     4            1
2     gene2         A     4            1
3     gene3         A     2            1
4     gene1         A     3            2
5     gene2         A     6            2
> mydat[1:5,1:2]
  gene.name condition
1     gene1         A
2     gene2         A
3     gene3         A
4     gene1         A
5     gene2         A
> 
```

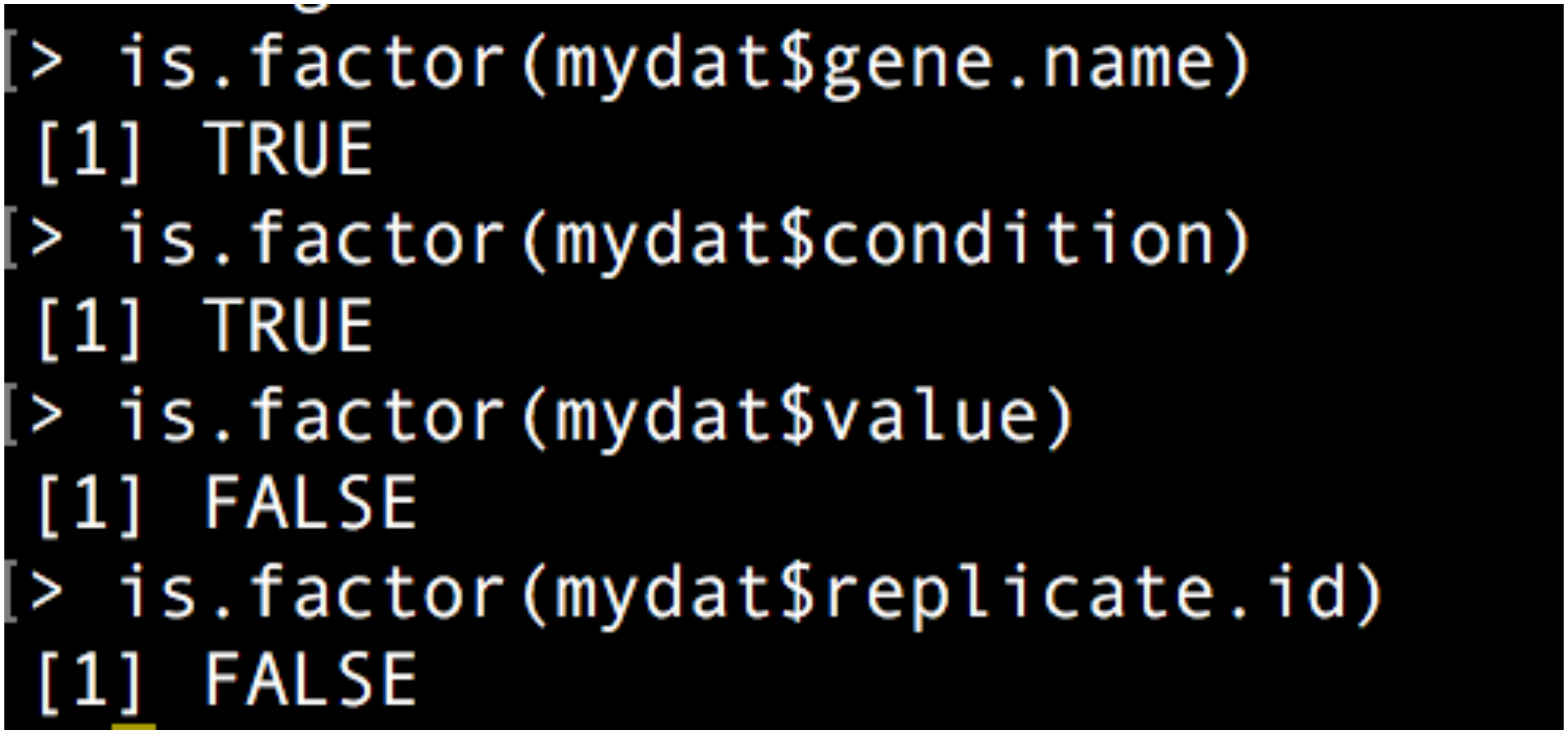
“Factors” in R

- Represent categorical variables – discrete and have a limited number of values
- Strings are often interpreted as factors by default
- Need caution when combining with arithmetic



```
[> mydat
   gene.name condition value replicate.id
1      gene1         A     4            1
2      gene2         A     4            1
3      gene3         A     2            1
4      gene1         A     3            2
5      gene2         A     6            2
6      gene3         A     9            2
7      gene1         B    10            1
8      gene2         B     5            1
9      gene3         B    10            1
10     gene1         B     8            2
11     gene2         B     3            2
12     gene3         B     0            2
[> mydat$condition
 [1] A A A A A A B B B B B B
Levels: A B
> 
```

“Factors” in R



```
> is.factor(mydat$gene.name)
[1] TRUE
> is.factor(mydat$condition)
[1] TRUE
> is.factor(mydat$value)
[1] FALSE
> is.factor(mydat$replicate.id)
[1] FALSE
```

Computers are stupid

Right

```
gene.name,condition,value,replicate.id
gene1,A,4,1
gene2,A,4,1
gene3,A,2,1
gene1,A,3,2
gene2,A,6,2
gene3,A,9,2
gene1,B,10,1
gene2,B,5,1
gene3,B,10,1
gene1,B,8,2
gene2,B,3,2
gene3,B,0,2
```

Wrong

```
gene.name,condition,value,replicate.id
gene1,A1,4,1
gene2,A1,4,1
gene3,A1,2,1
gene1,A2,3,2
gene2,A2,6,2
gene3,A,29,2
gene1,B1,10,1
gene2,B1,5,1
gene3,B1,10,1
gene1,B2,8,2
gene2,B2,3,2
gene3,B2,0,2
```

Computers are stupid

Right

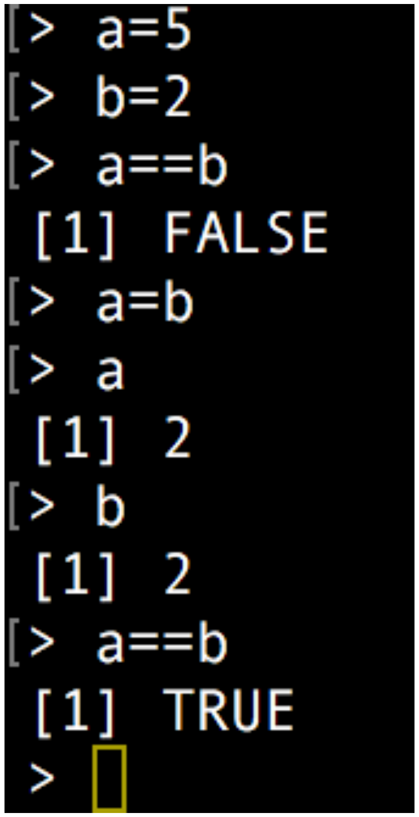
```
gene.name,condition,value,replicate.id
gene1,A,4,1
gene2,A,4,1
gene3,A,2,1
gene1,A,3,2
gene2,A,6,2
gene3,A,9,2
gene1,B,10,1
gene2,B,5,1
gene3,B,10,1
gene1,B,8,2
gene2,B,3,2
gene3,B,0,2
```

Wrong

```
gene.name,condition,value,replicate.id
gene1,A1,4,1
gene2,A1,4,1
gene3,A1,2,1
gene1,A2,3,2
gene2,A2,6,2
gene3,A,29,2
gene1,B1,10,1
gene2,B1,5,1
gene3,B1,10,1
gene1,B2,8,2
gene2,B2,3,2
gene3,B2,0,2
```

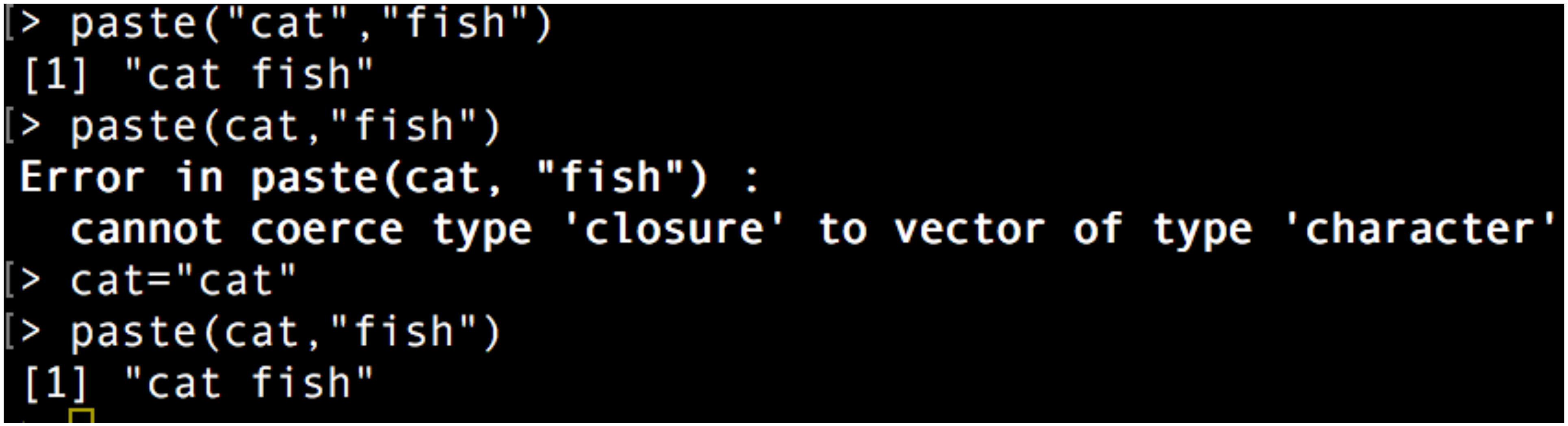
Computers are stupid

a = b vs. a == b


```
[> a=5
[> b=2
[> a==b
 [1] FALSE
[> a=b
[> a
 [1] 2
[> b
 [1] 2
[> a==b
 [1] TRUE
 > 
```

Computers are stupid

cat vs. “cat”



```
> paste("cat","fish")
[1] "cat fish"
> paste(cat,"fish")
Error in paste(cat, "fish") :
  cannot coerce type 'closure' to vector of type 'character'
> cat="cat"
> paste(cat,"fish")
[1] "cat fish"
```

Documentation is smart

?t.test

help(t.test)

help.search(“t-test”)

Google (stackoverflow)

There are packages for almost everything

Examples:

- plyr/dplyr for combining datasets
- biomart for looking up annotations
- flowCore/flowViz for flow cytometry data

There are packages for almost everything

Examples:

- plyr/dplyr for combining datasets
- **biomart for looking up annotations**
- flowCore/flowViz for flow cytometry data

Before:
FBgn0030482

After:
FBgn0030482; Branched-chain-amino-acid aminotransferase [Source:UniProtKB/TrEMBL;Acc:Q9VYD5]

Outline

- Data import and management in R
- **Overview of machine learning**
- Common machine learning methods
- Applications of machine learning in biology

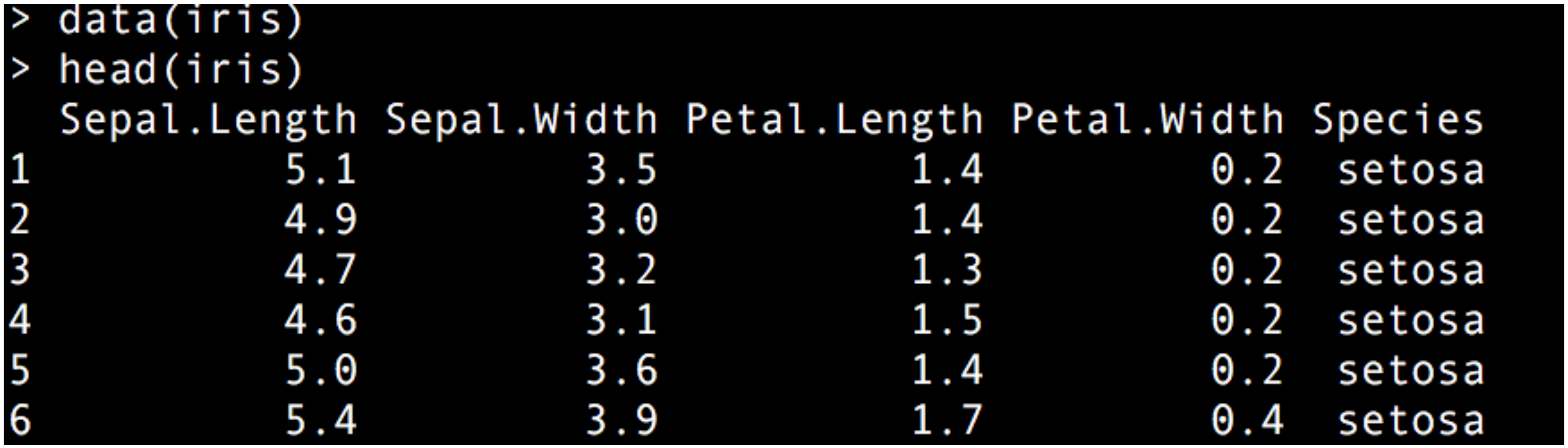
Machine learning

“[F]ield of study that gives computers the ability to learn without being explicitly programmed”

--Arthur Samuel, 1959

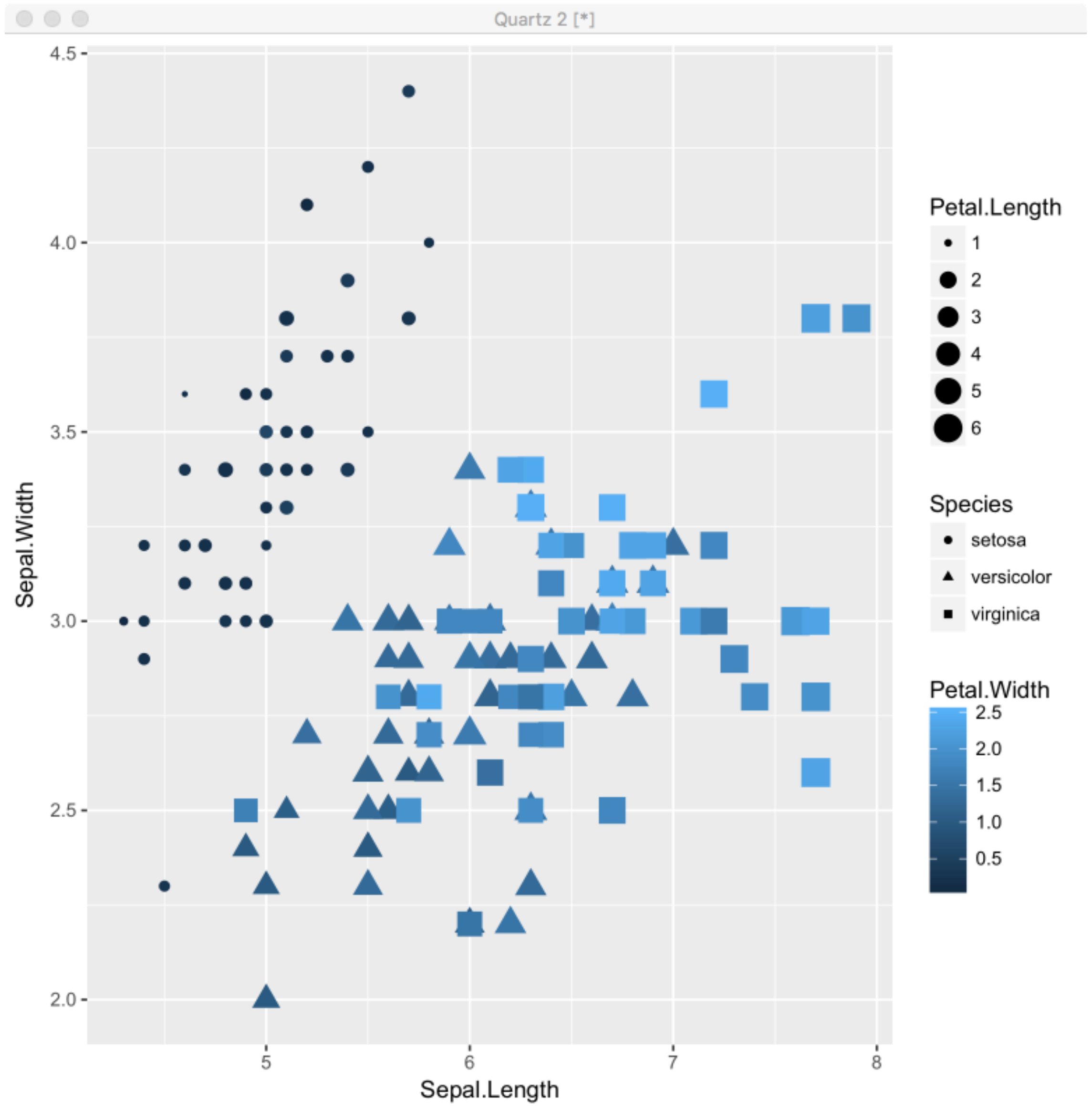
We want the computer to derive insight from a data set and either tell us about it or use it for a future problem

Example dataset: iris

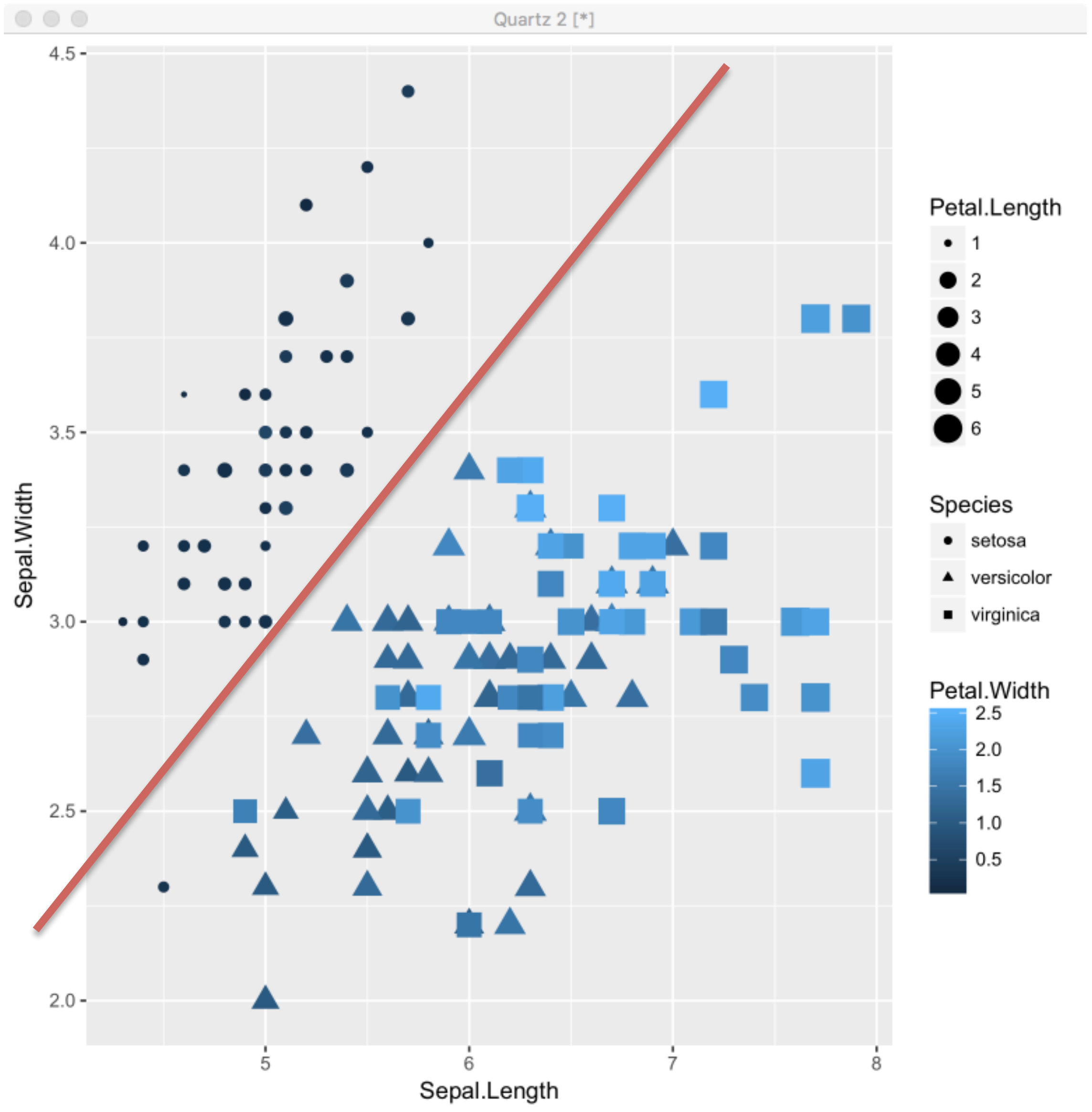

```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Example dataset: iris

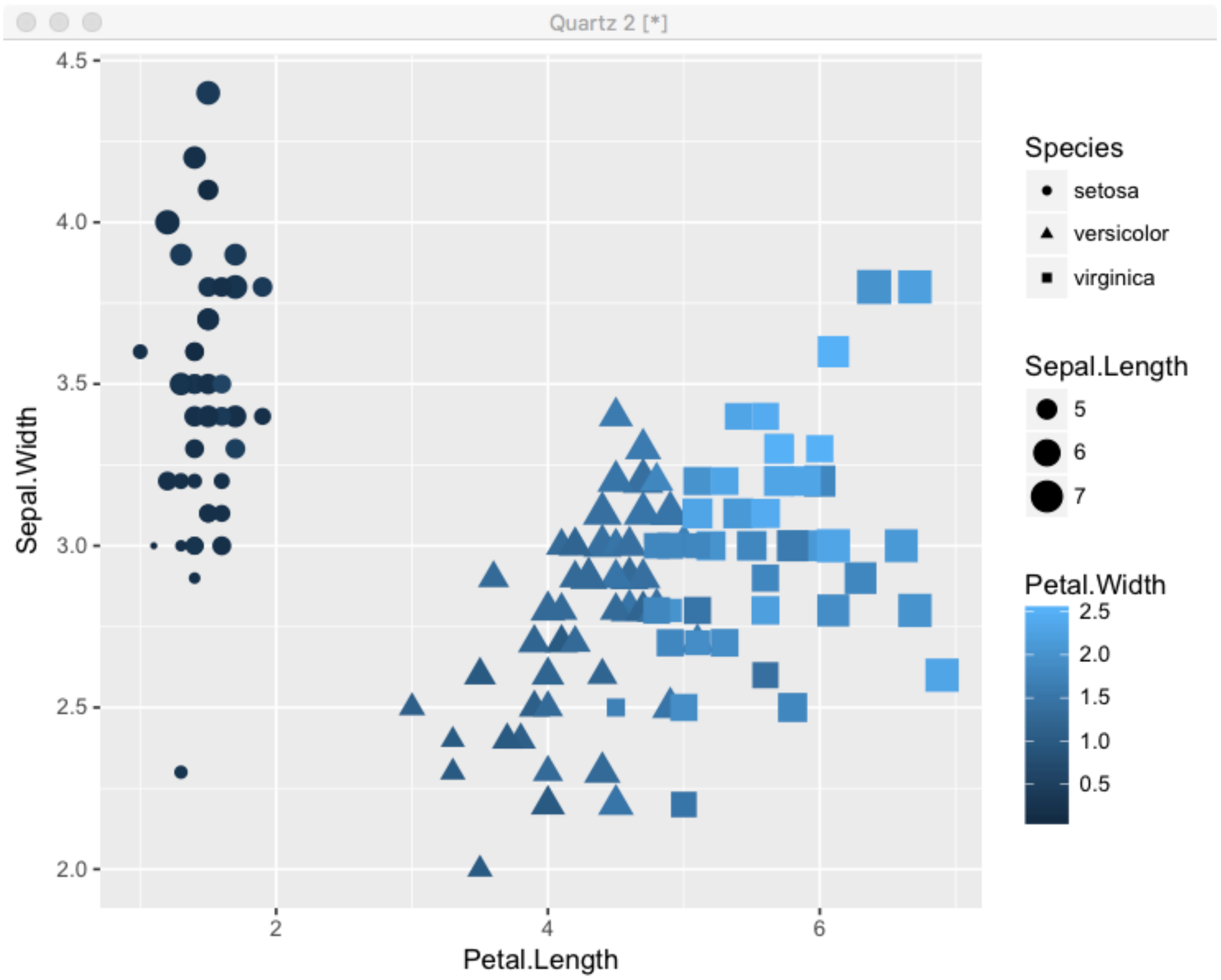
```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length, color=Petal.Width, shape=Species)) + geom_point()
```



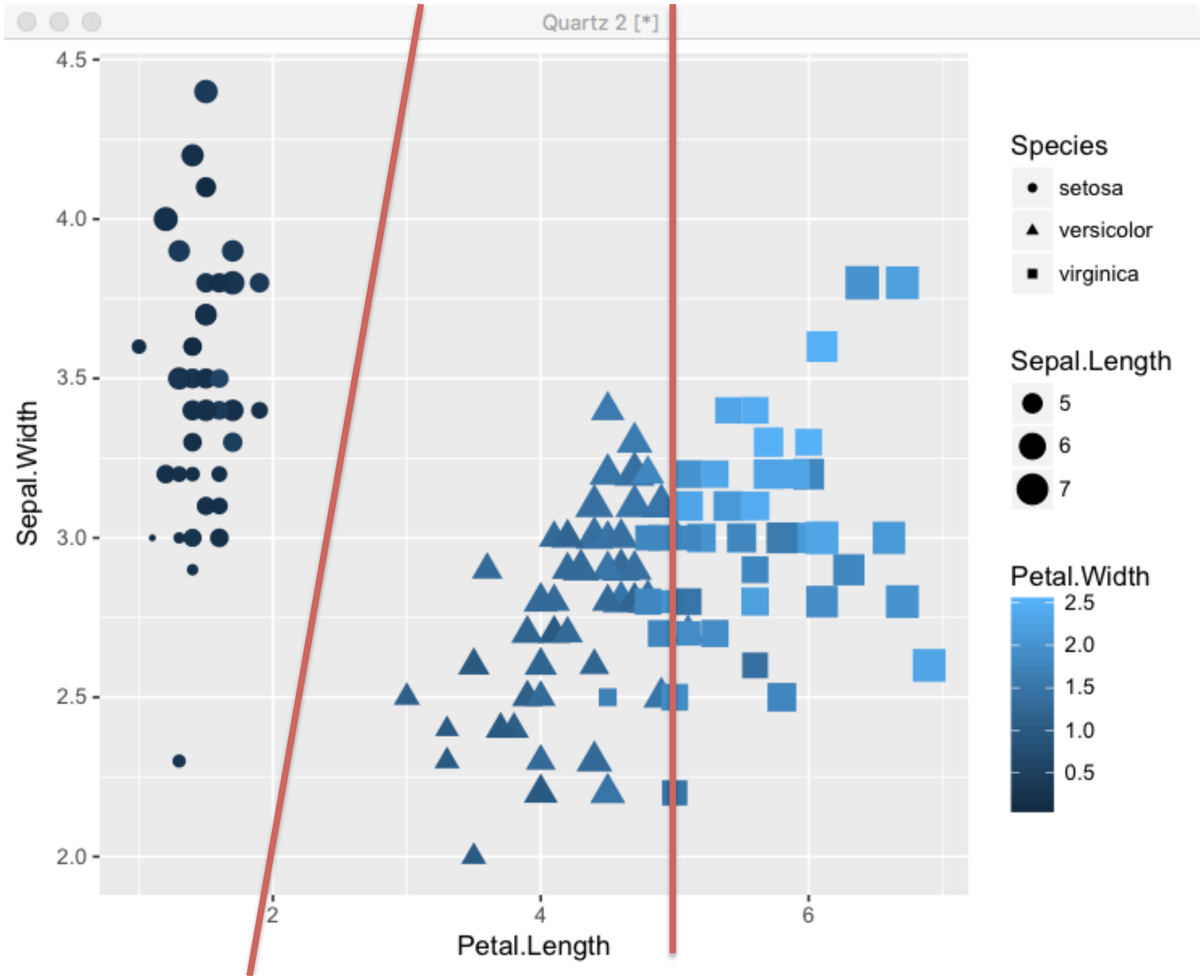
Example dataset: iris



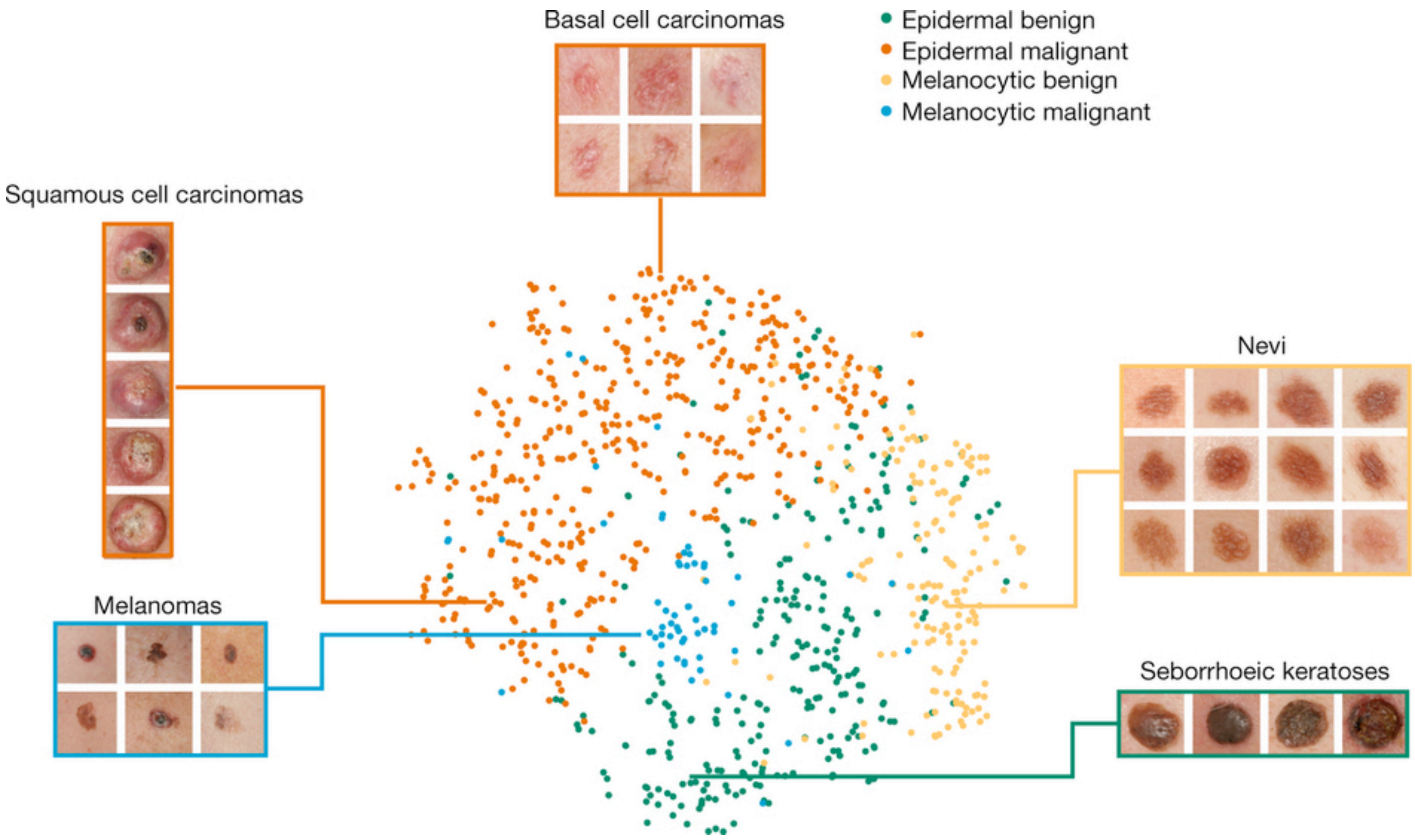
Example dataset: iris



Example dataset: iris



Skin lesion classification



Nature 542, 115–118 (02 February 2017)

Common machine learning questions

- Given values of observables (e.g., sepal length), what species did a specimen come from?
- Which characteristics are most useful in figuring out which species a specimen came from?
- What does an average member of the setosa species look like?
- If I don’t already know the answer, how many species are present, and which specimens come from which species?

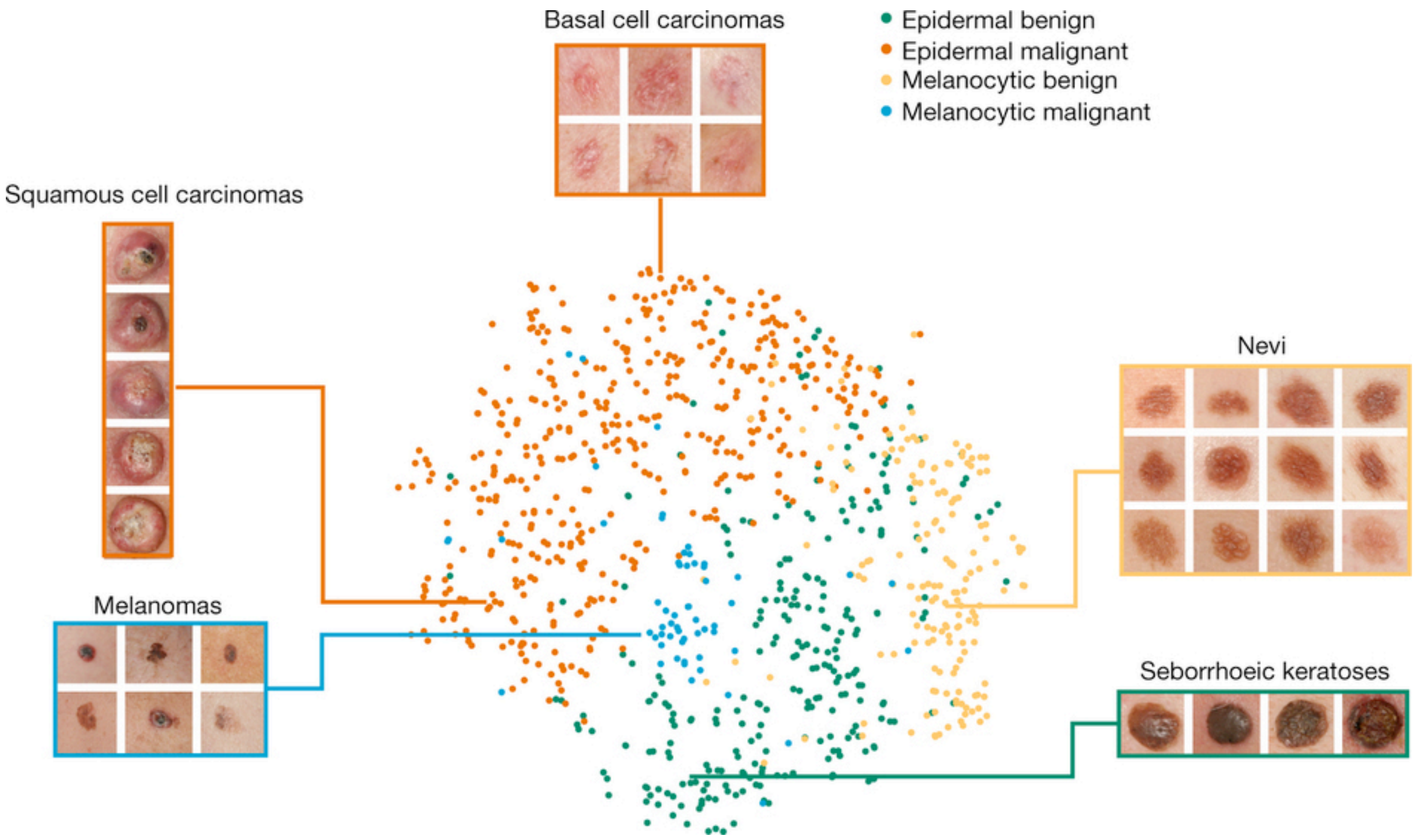
Types of machine learning tasks

- Supervised: We can tell the algorithm correct answers on a *training set*, and then expect it to work from there
- Unsupervised: We provide no prior information about the data set
- Reinforced: We provide feedback over the course of algorithm optimization

Types of machine learning tasks

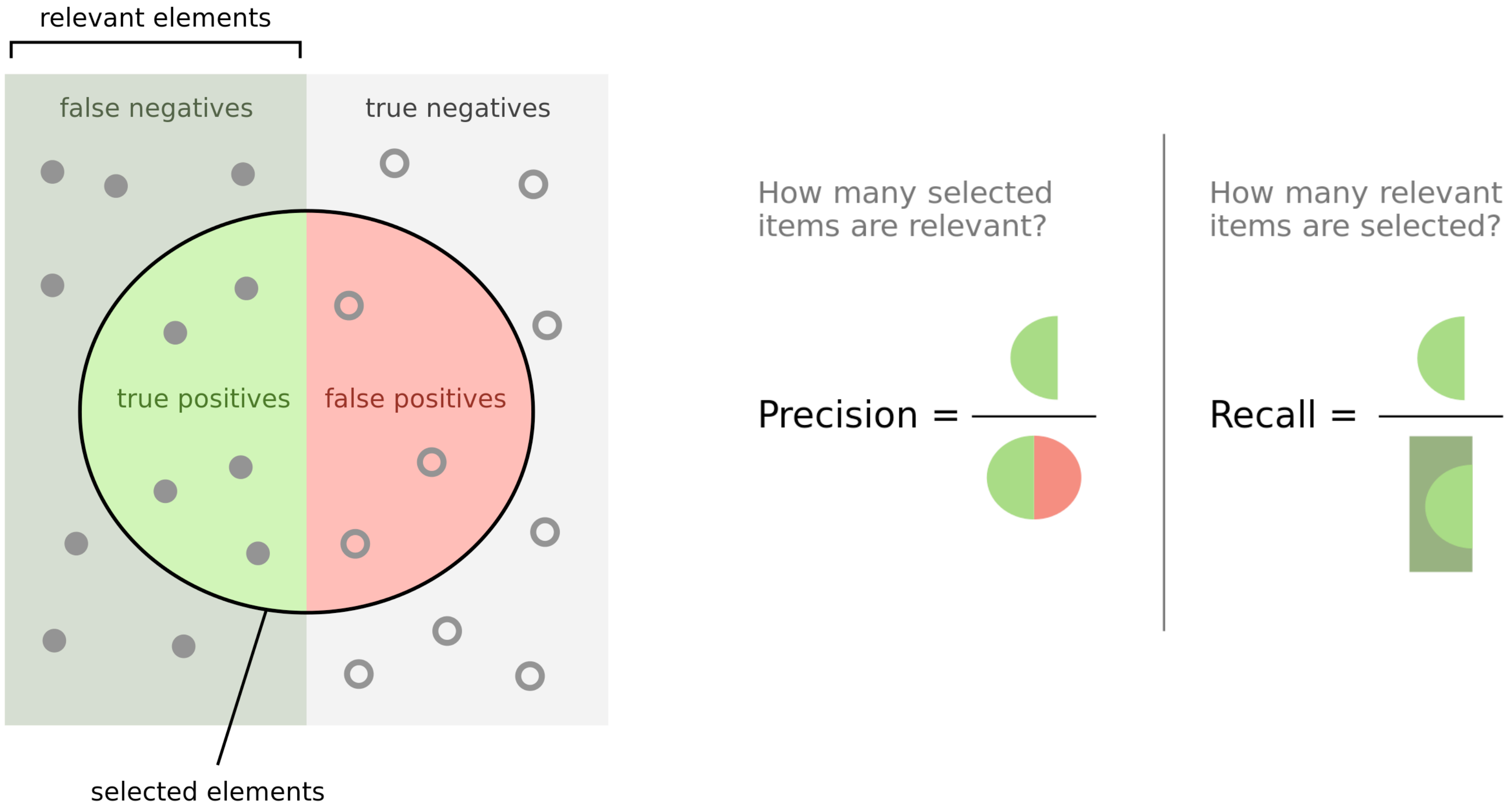
- Classification: Which category does a sample fall into (supervised)
- Clustering: How many categories are there and which one does each sample fall into (unsupervised)
- Dimensionality reduction: How can I more simply visualize a data distribution?

Skin lesion classification



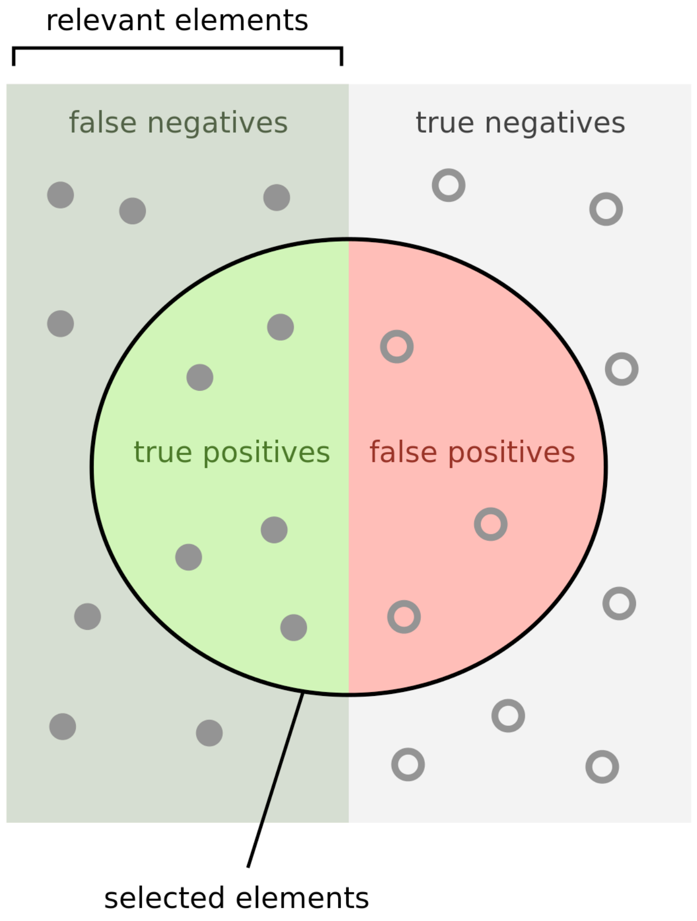
Nature 542, 115–118 (02 February 2017)

Performance evaluation



(Image from wikipedia user Walber)

Performance evaluation



Matthews correlation coefficient:

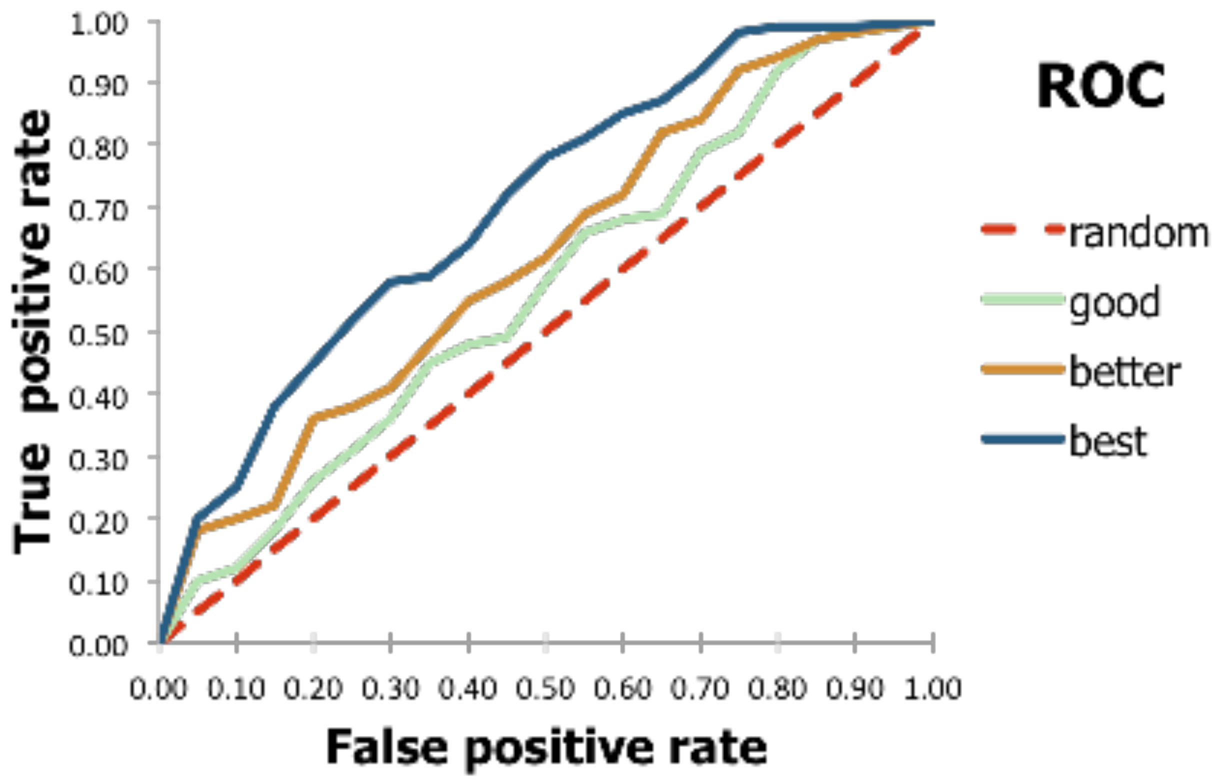
$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

F-measure:

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

(Image from wikipedia user Walber)

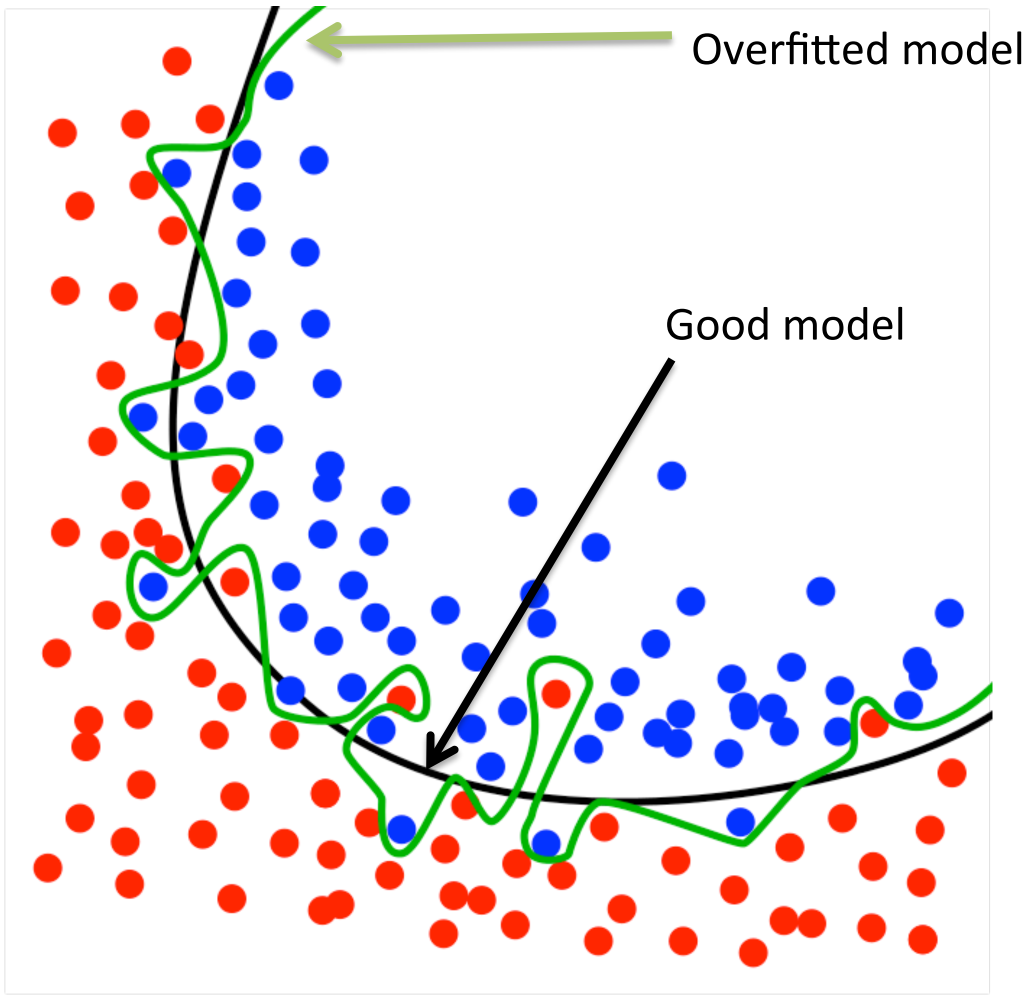
Performance evaluation



ROC: Receiver operating characteristic
AUC: Area under (ROC) curve

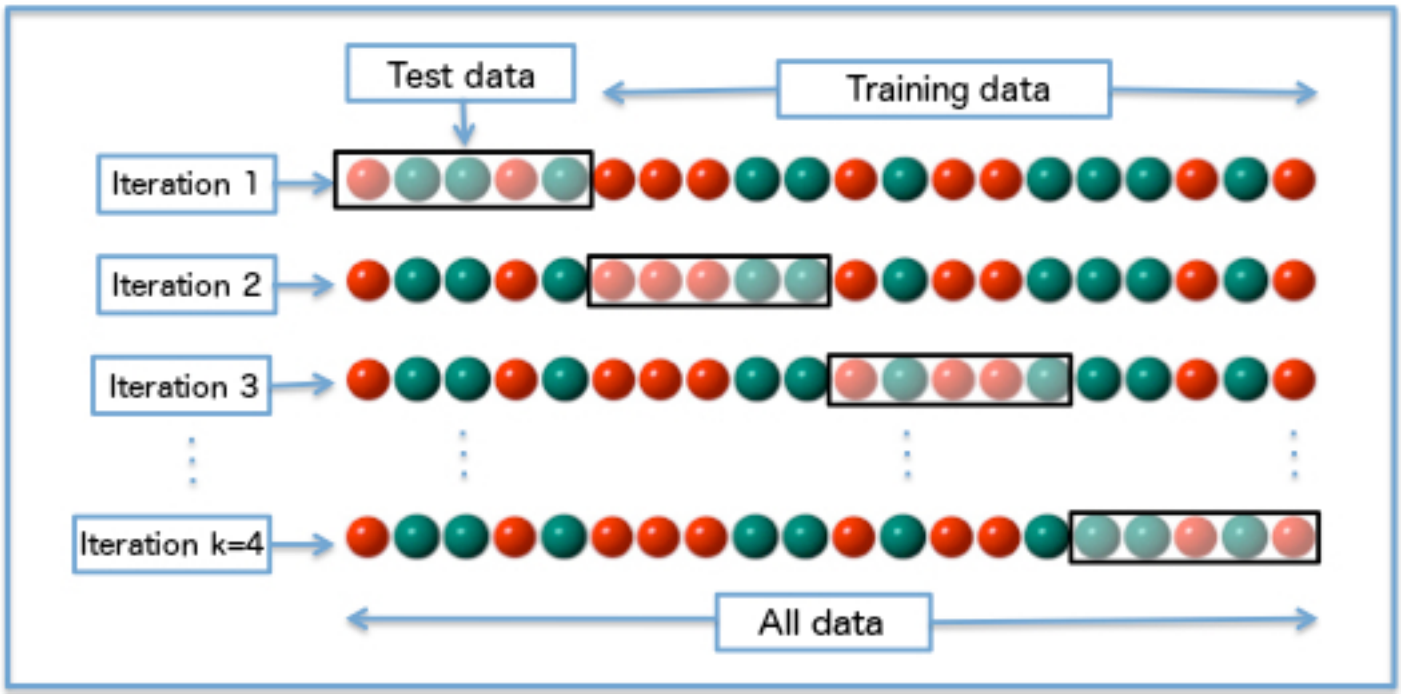
(Image from OpenEye Scientific)

Avoiding overfitting



(Image from wikipedia user Chabacano under GFDL)

Avoiding overfitting



4-fold cross-validation

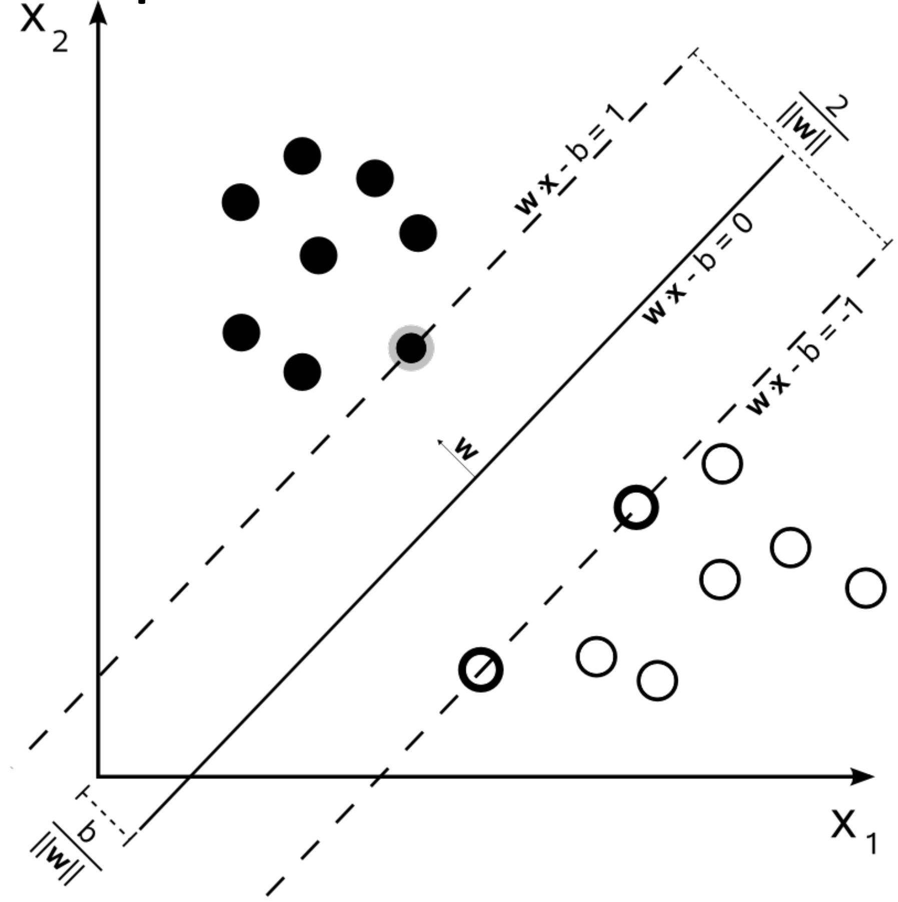
(Image from wikipedia user Fabian Flock)

Outline

- Data import and management in R
- Overview of machine learning
- **Common machine learning methods**
- Applications of machine learning in biology

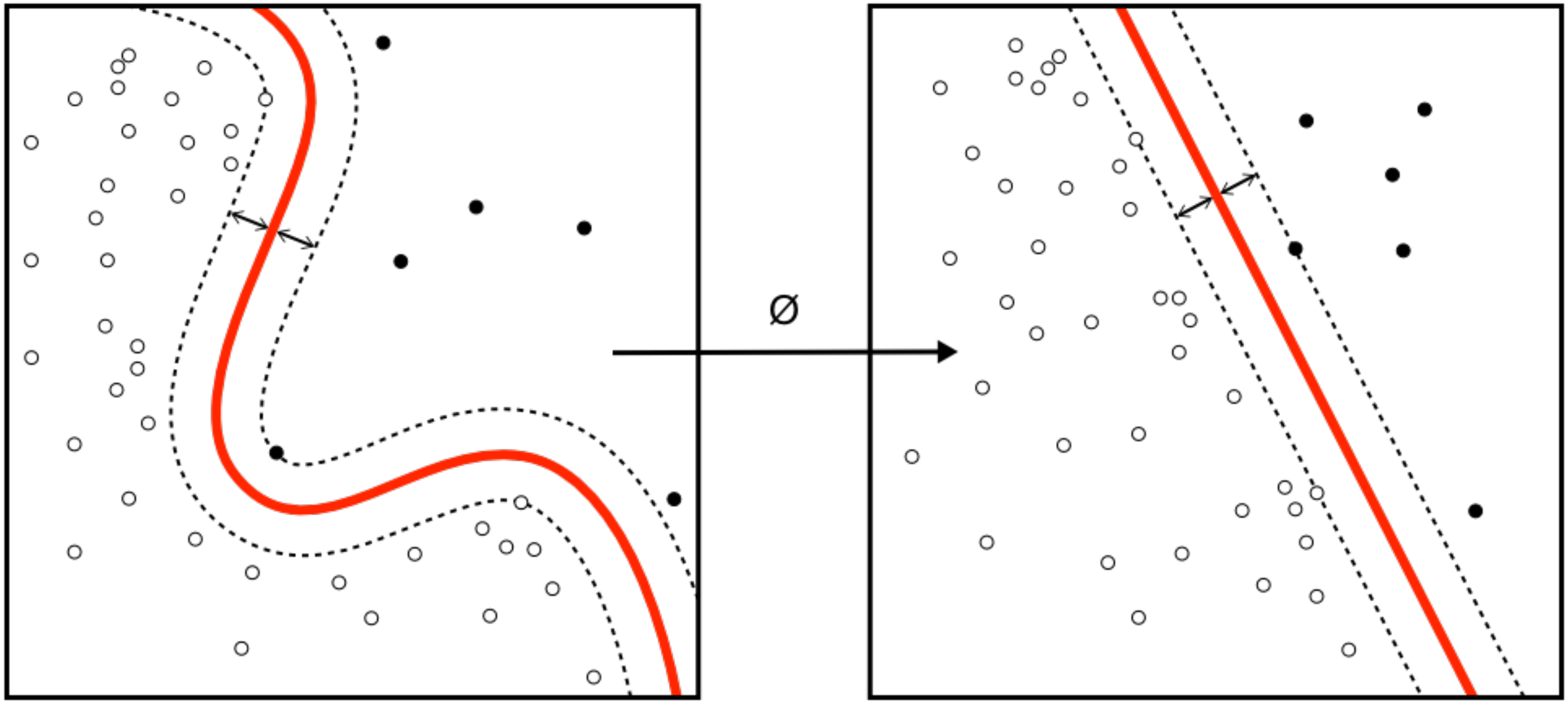
Support vector machines (SVMs)

Supervised learning: Find a partition that maximizes separation between sets

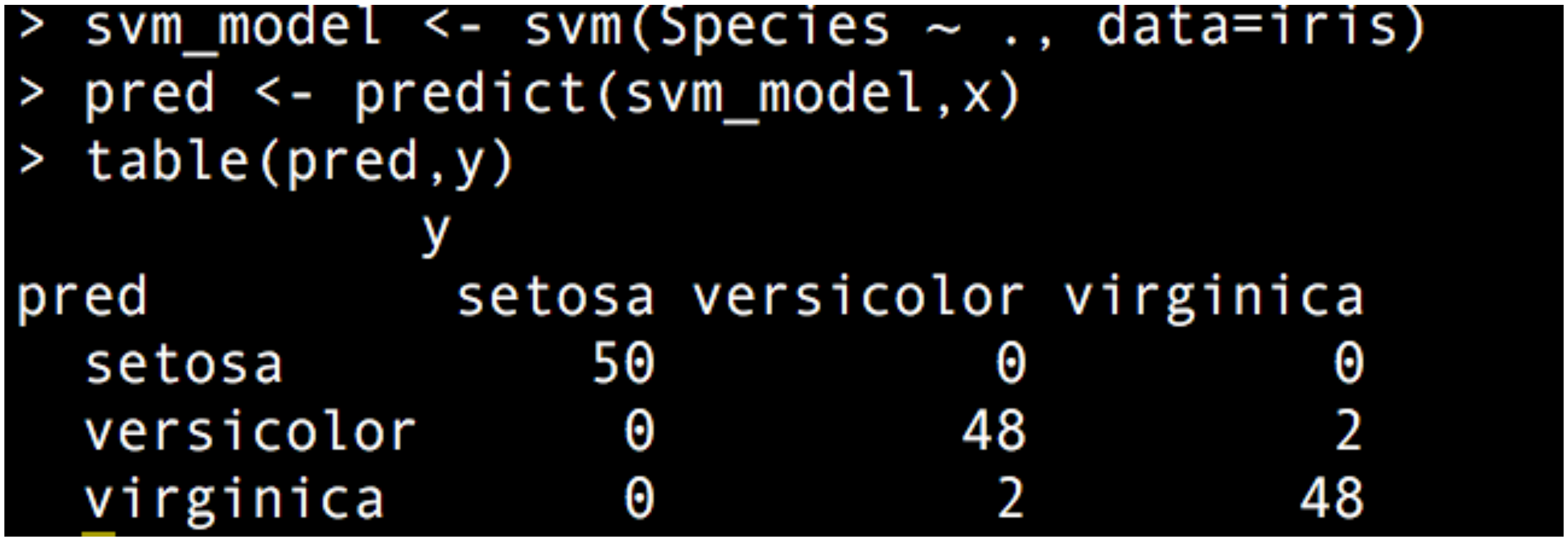

x_2
x_1
$\mathbf{w}\cdot\mathbf{x} - b = 1$
$\mathbf{w}\cdot\mathbf{x} - b = 0$
$\mathbf{w}\cdot\mathbf{x} - b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

Support vector machines (SVMs)

Supervised learning: Find a partition that maximizes separation between sets

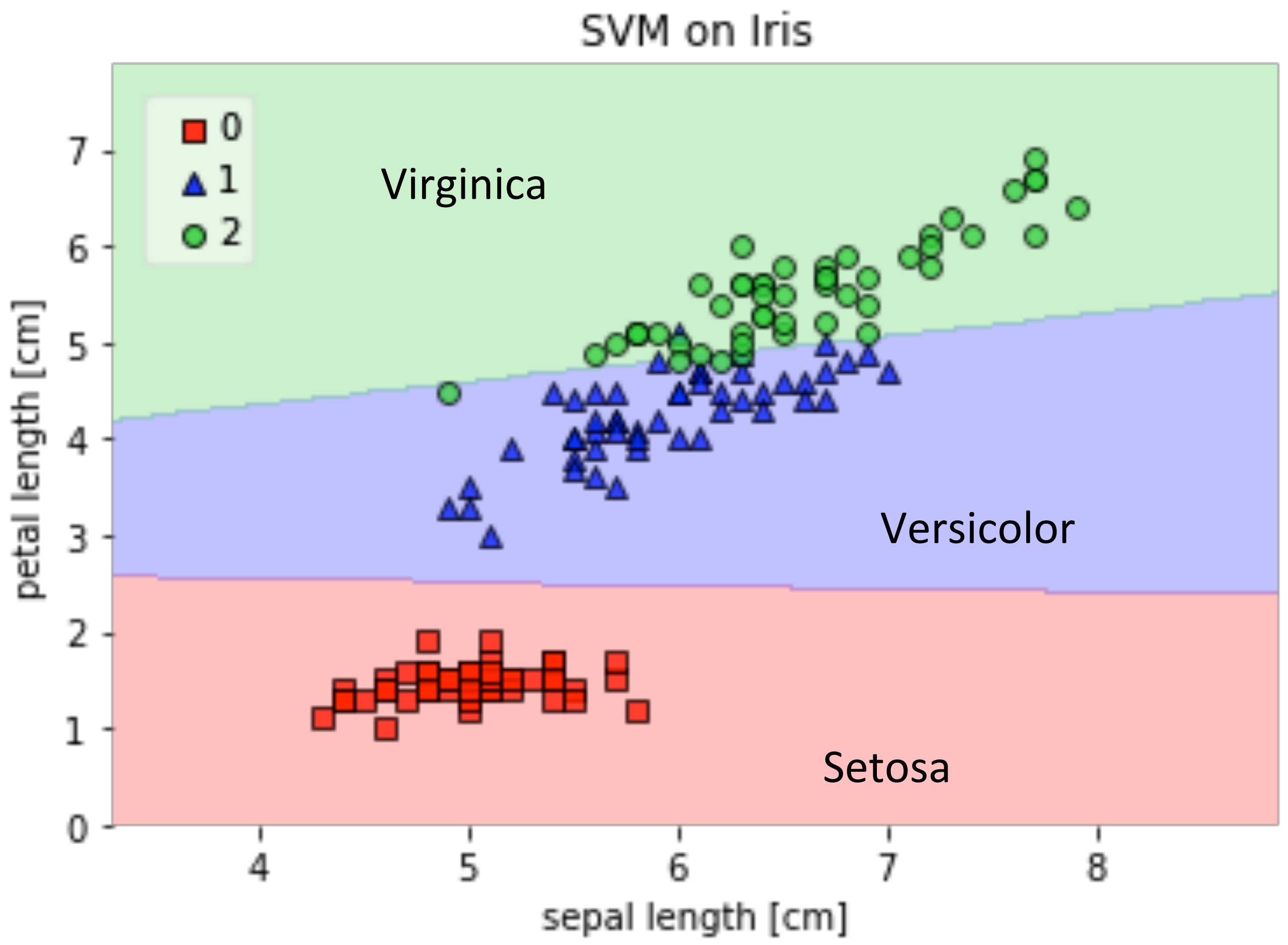


Applying an SVM to the iris dataset



```
> svm_model <- svm(Species ~ ., data=iris)
> pred <- predict(svm_model,x)
> table(pred,y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
```

Applying an SVM to the iris dataset

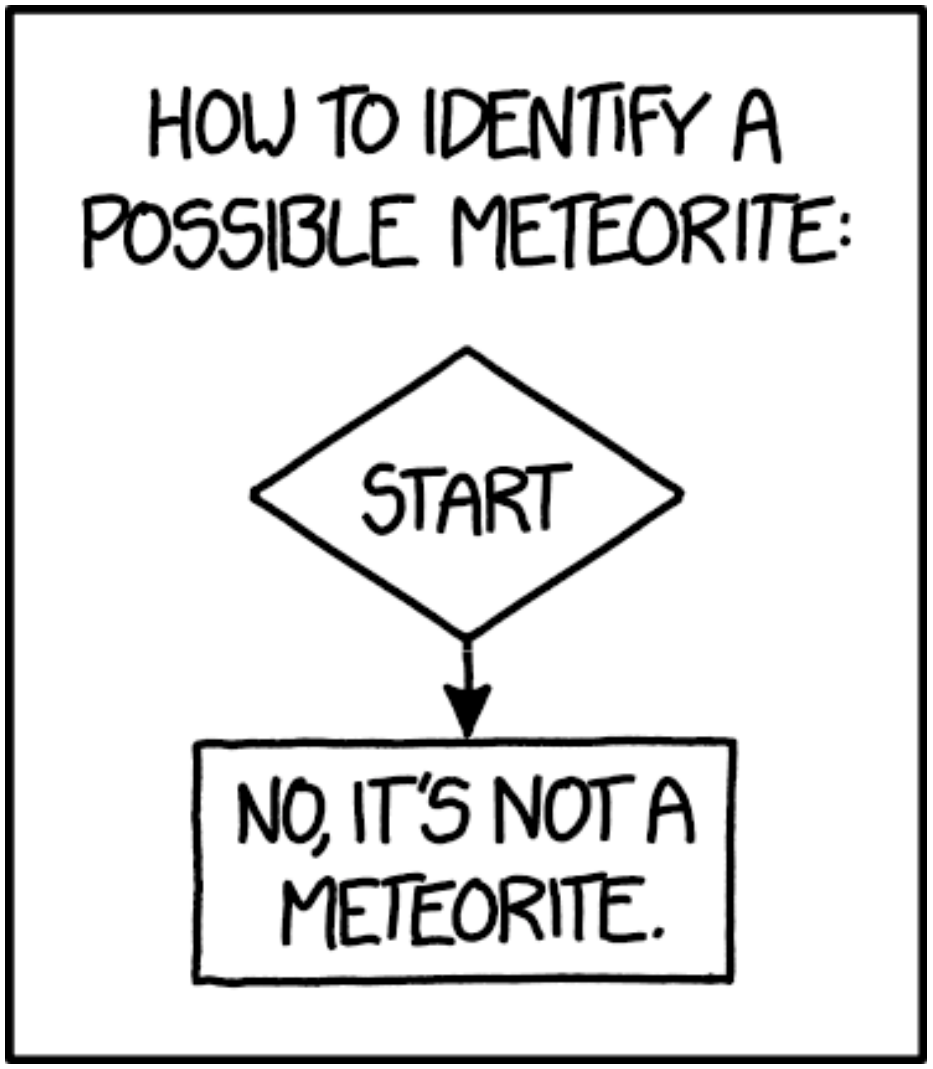


(Image from http://rasbt.github.io/mlxtend/user_guide/plotting/plot_decision_regions/)

Random forests

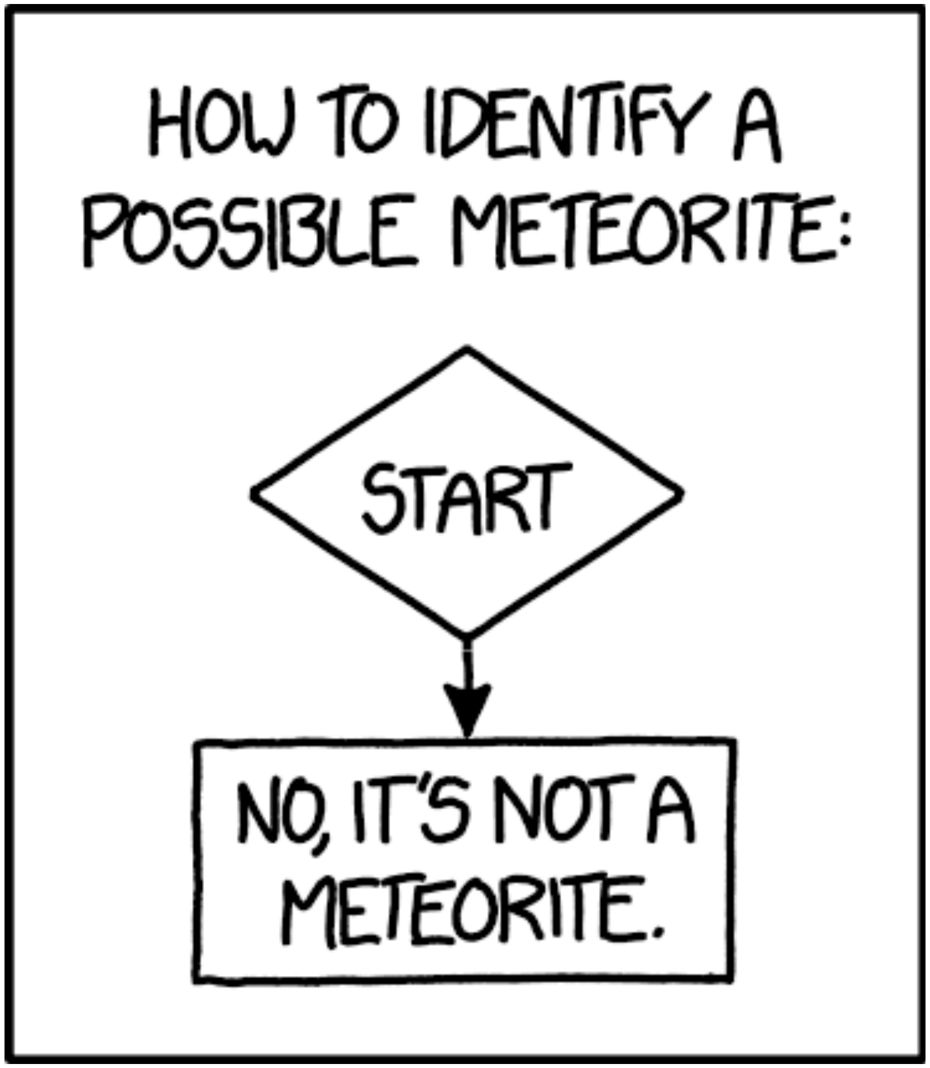
Supervised learning: Find a “forest” of decision trees to optimize classification performance

Decision trees

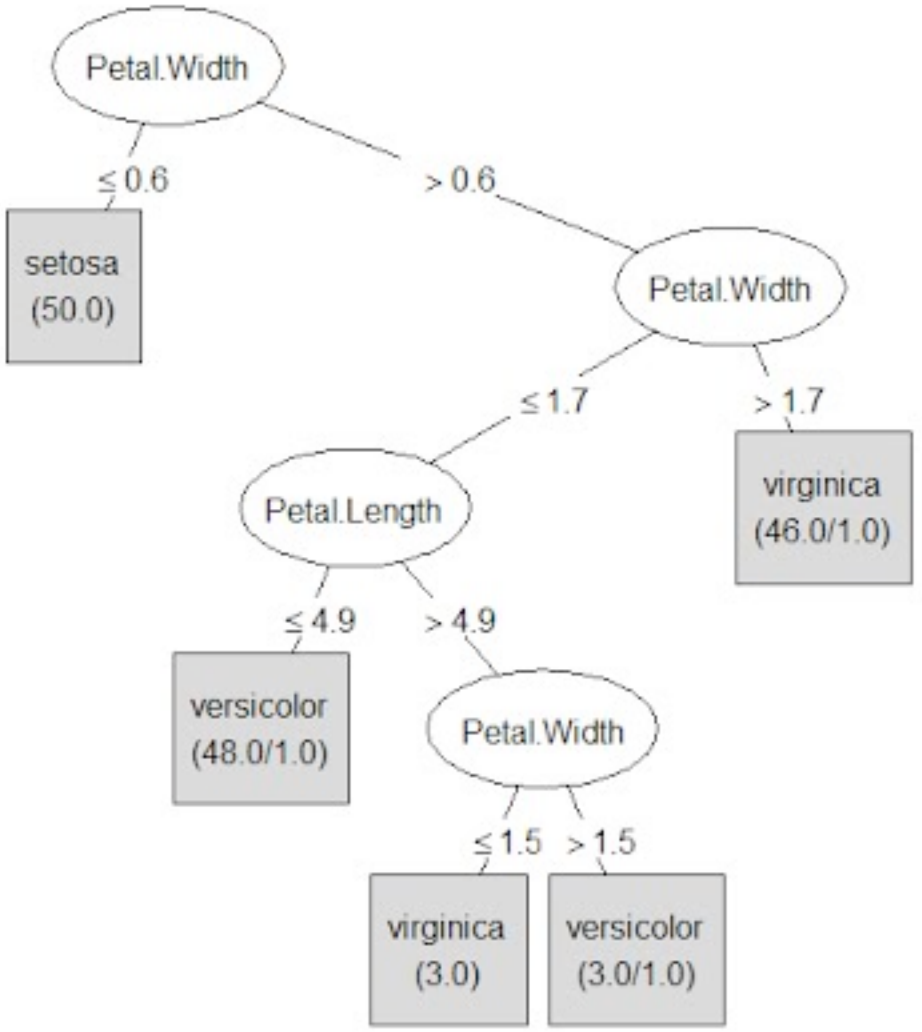

HOW TO IDENTIFY A POSSIBLE METEORITE:
START
NO, IT'S NOT A METEORITE.

https://xkcd.com/1723/

Decision trees


HOW TO IDENTIFY A POSSIBLE METEORITE:
START
NO, IT'S NOT A METEORITE.

https://xkcd.com/1723/



http://data-mining.business-intelligence.uoc.edu/home/j48-decision-tree

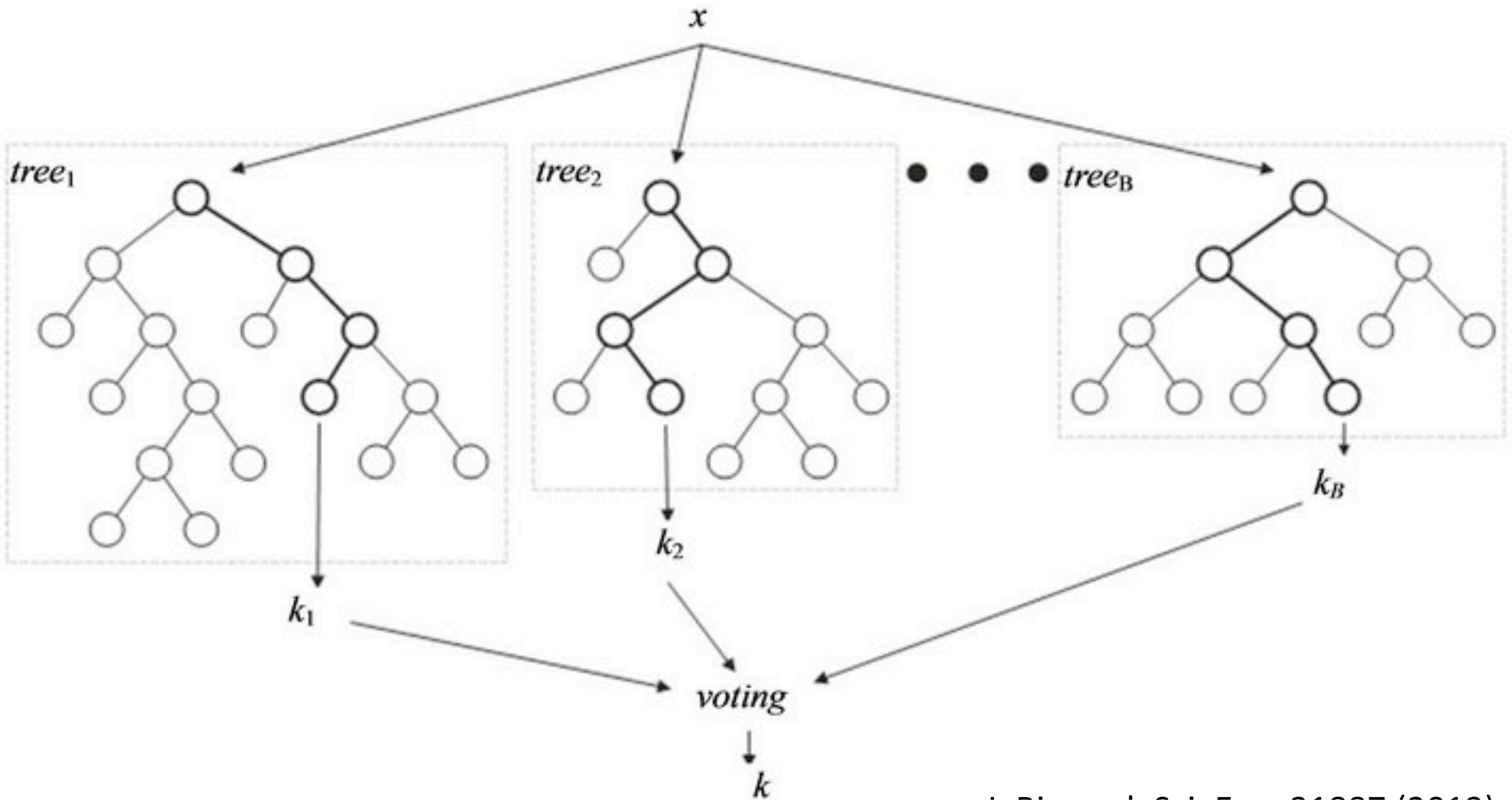
Decision trees

Of all the well-known learning methods, decision trees come closest to
meeting the requirements for serving as an off-the-shelf procedure for data
mining. They are relatively fast to construct and they produce interpretable
models (if the trees are small)… and they are immune to the effects of predictor outliers.
They perform internal feature selection as an integral part
of the procedure. They are thereby resistant, if not completely immune,
to the inclusion of many irrelevant predictor variables. These properties of
decision trees are largely the reason that they have emerged as the most
popular learning method for data mining. Trees have one aspect that prevents them from
being the ideal tool for predictive learning, namely inaccuracy. They seldom provide
predictive accuracy comparable to the best that can be achieved with the data at hand.

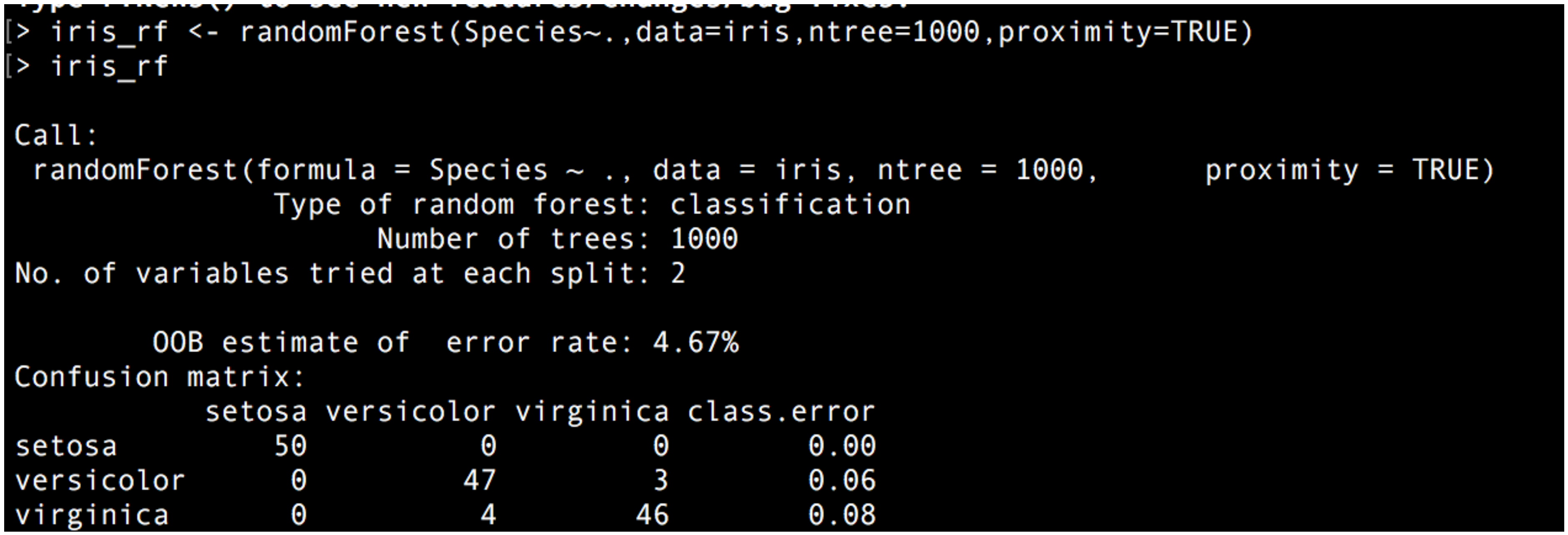
--Hastie et al., *The Elements of Statistical Learning*

Random forests

Supervised learning: Find a "forest" of decision trees to optimize classification performance

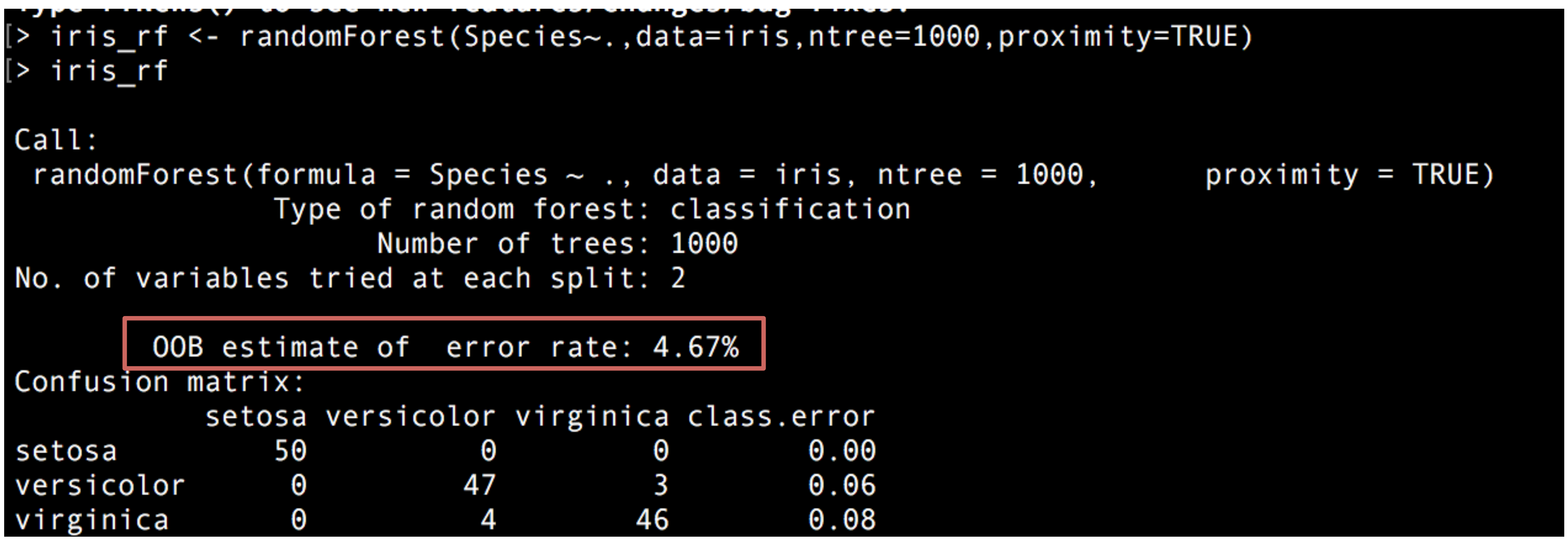


Example application to iris data



```
> iris_rf <- randomForest(Species~.,data=iris,ntree=1000,proximity=TRUE)
> iris_rf

Call:
 randomForest(formula = Species ~ ., data = iris, ntree = 1000,      proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.67%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          4        46        0.08
```

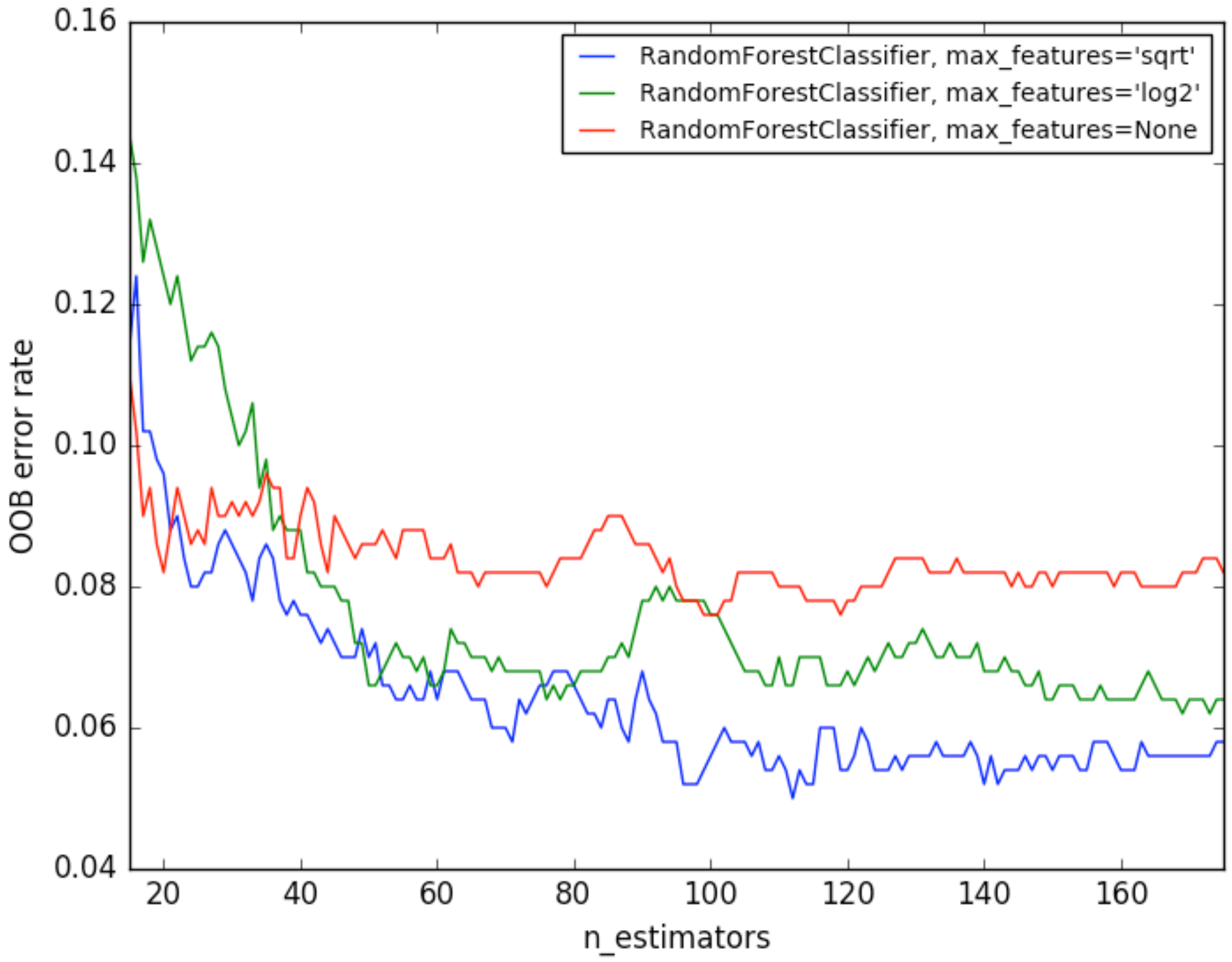
Example application to iris data



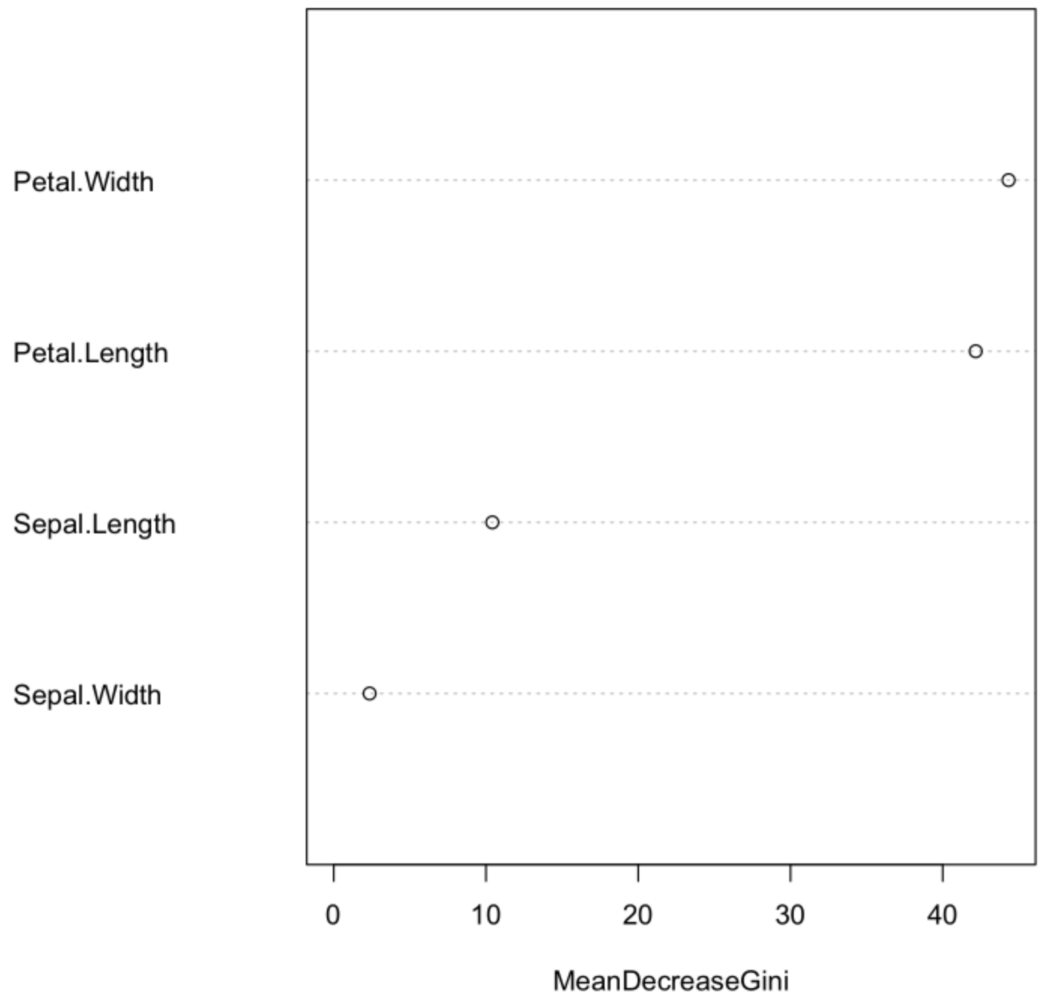
```
[> iris_rf <- randomForest(Species~.,data=iris,ntree=1000,proximity=TRUE)
[> iris_rf

Call:
 randomForest(formula = Species ~ ., data = iris, ntree = 1000,      proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.67%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          4        46        0.08
```

OOB (out of bag estimate): Estimates error rates based on classifications ignoring certain training data (similar to cross-validation)

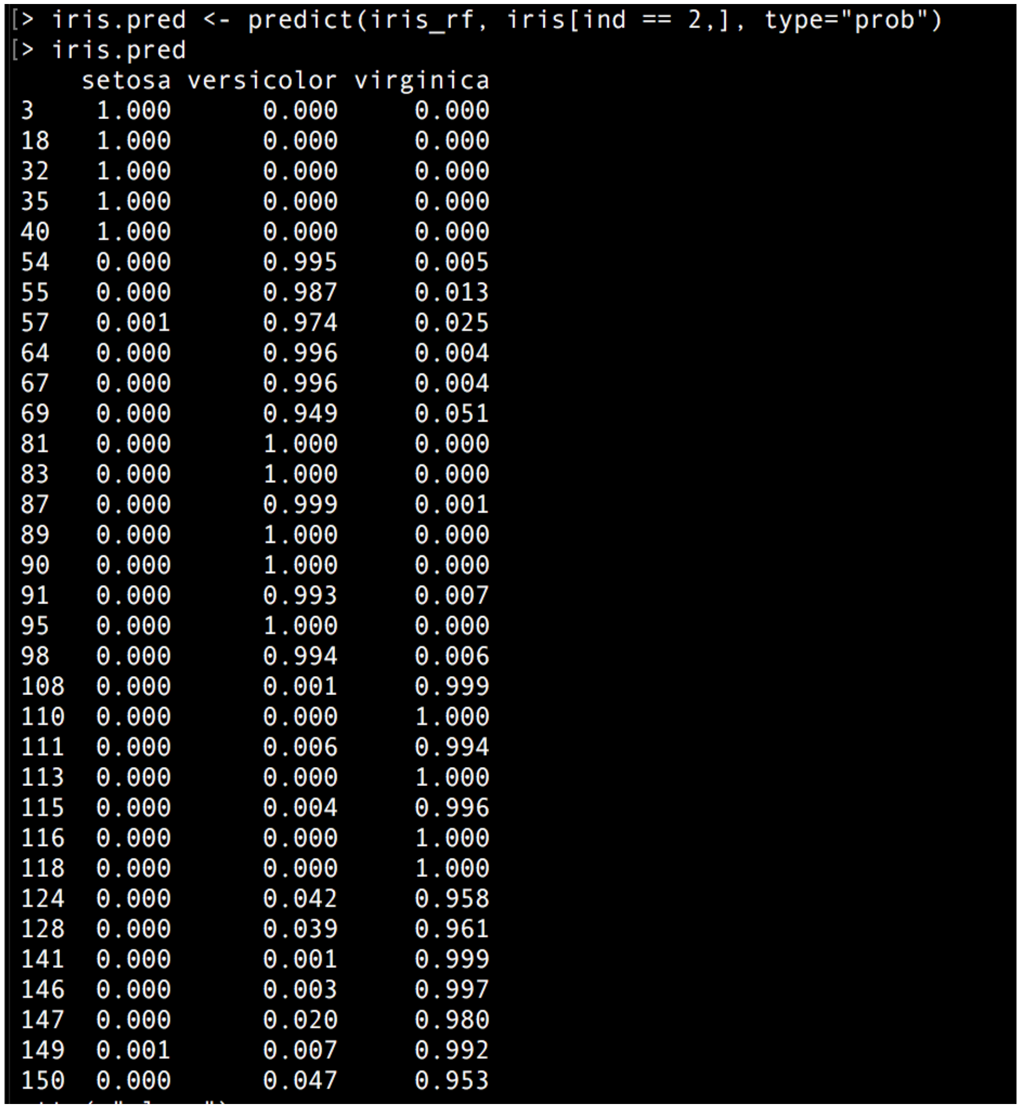
Determining the number of trees



Identifying important variables

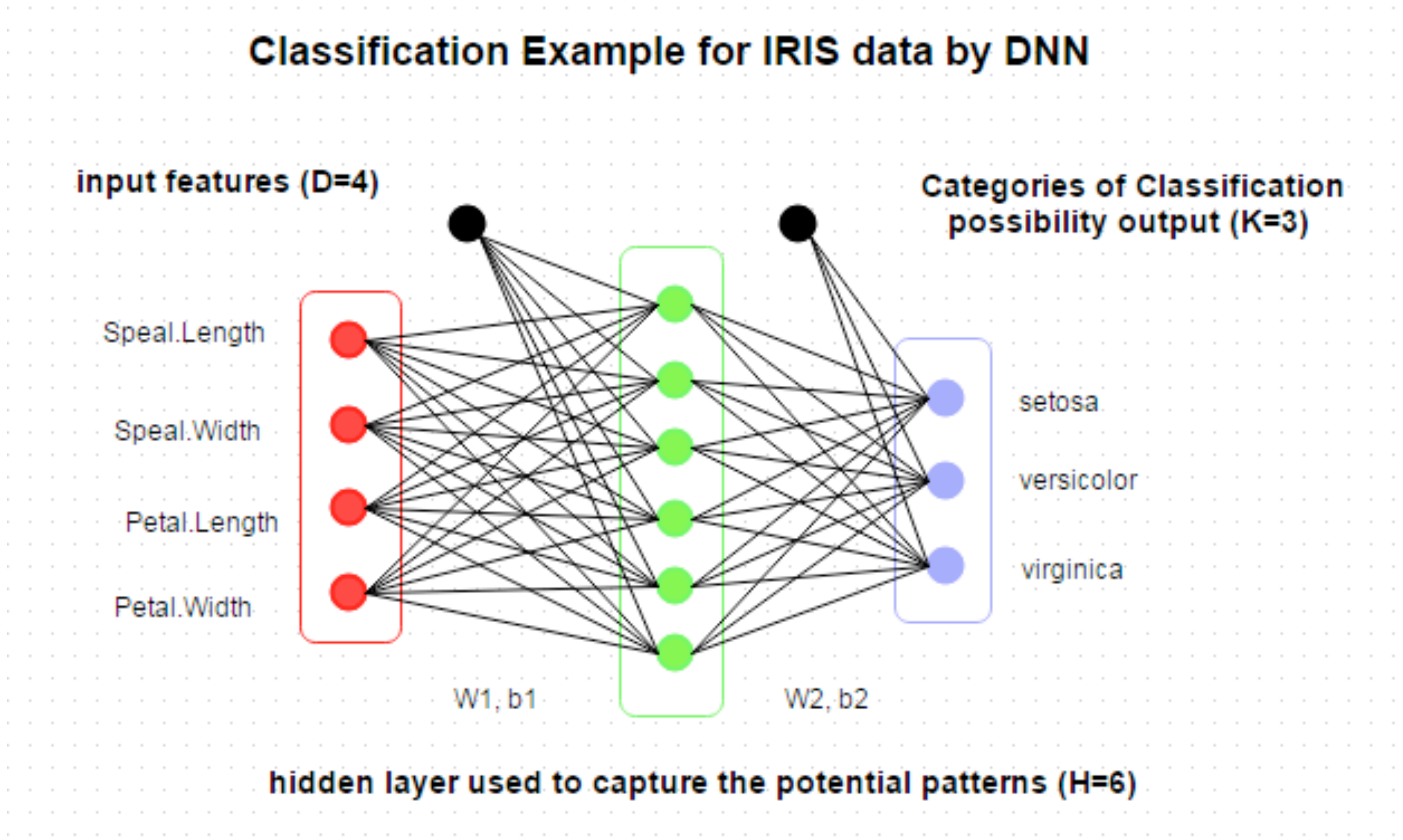


Predictions for new cases



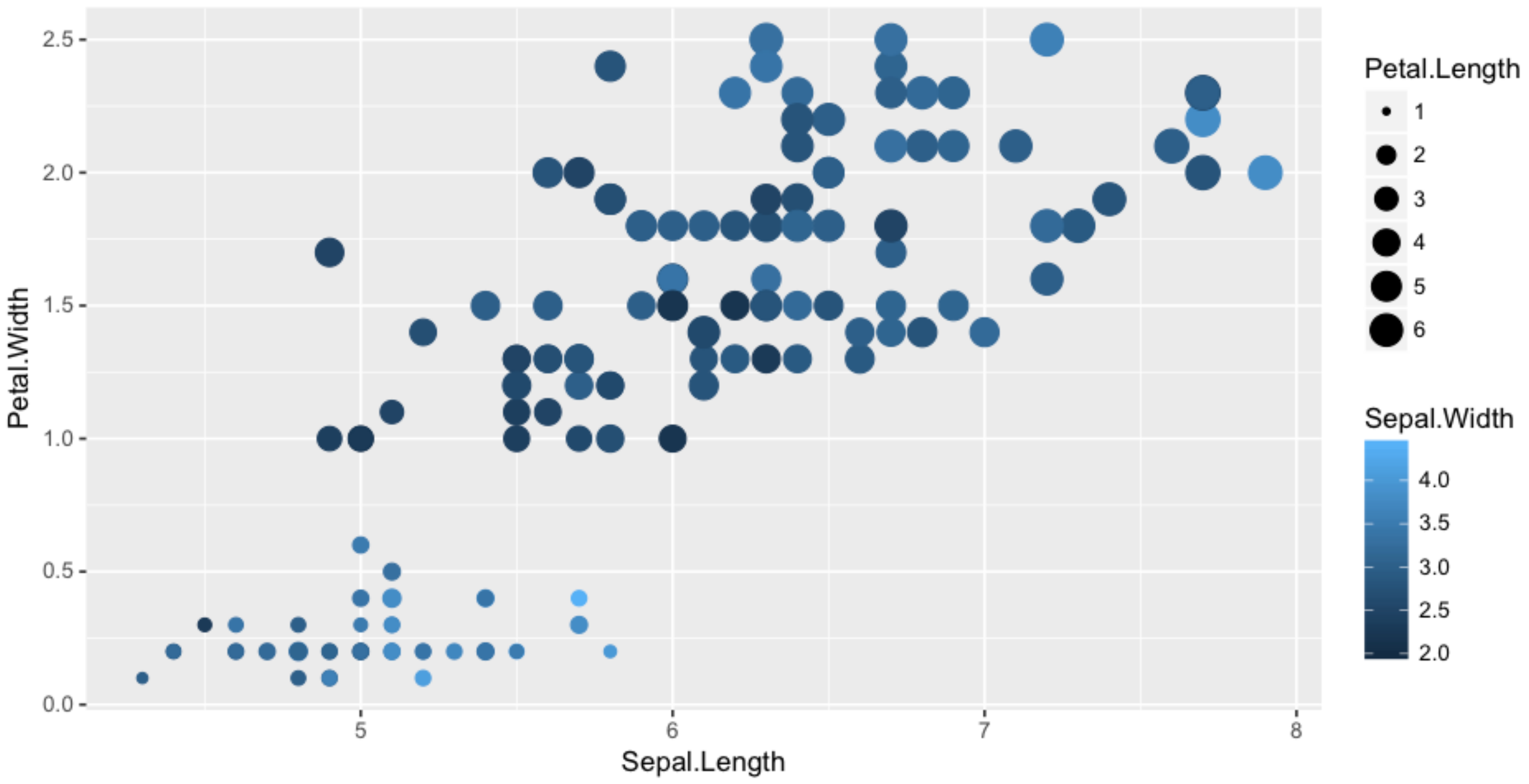
```
[> iris.pred <- predict(iris_rf, iris[ind == 2,], type="prob")
[> iris.pred
    setosa versicolor virginica
3    1.000      0.000     0.000
18   1.000      0.000     0.000
32   1.000      0.000     0.000
35   1.000      0.000     0.000
40   1.000      0.000     0.000
54   0.000      0.995     0.005
55   0.000      0.987     0.013
57   0.001      0.974     0.025
64   0.000      0.996     0.004
67   0.000      0.996     0.004
69   0.000      0.949     0.051
81   0.000      1.000     0.000
83   0.000      1.000     0.000
87   0.000      0.999     0.001
89   0.000      1.000     0.000
90   0.000      1.000     0.000
91   0.000      0.993     0.007
95   0.000      1.000     0.000
98   0.000      0.994     0.006
108  0.000      0.001     0.999
110  0.000      0.000     1.000
111  0.000      0.006     0.994
113  0.000      0.000     1.000
115  0.000      0.004     0.996
116  0.000      0.000     1.000
118  0.000      0.000     1.000
124  0.000      0.042     0.958
128  0.000      0.039     0.961
141  0.000      0.001     0.999
146  0.000      0.003     0.997
147  0.000      0.020     0.980
149  0.001      0.007     0.992
150  0.000      0.047     0.953
```

Deep learning/deep neural networks


Classification Example for IRIS data by DNN
input features (D=4)
Categories of Classification possibility output (K=3)
Speal.Length
Speal.Width
Petal.Length
Petal.Width
setosa
versicolor
virginica
W1, b1
W2, b2
hidden layer used to capture the potential patterns (H=6)

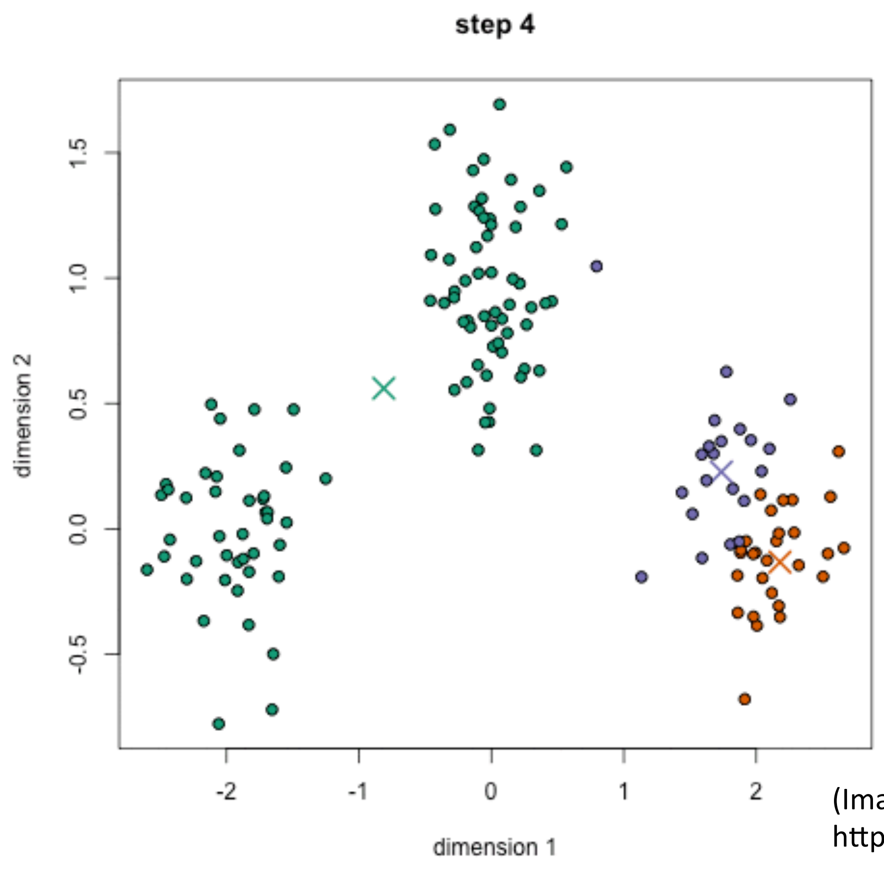
(From http://www.parallelr.com/r-deep-neural-network-from-scratch/iris_network/)

Unsupervised clustering: When we don't have a training set



K-means clustering

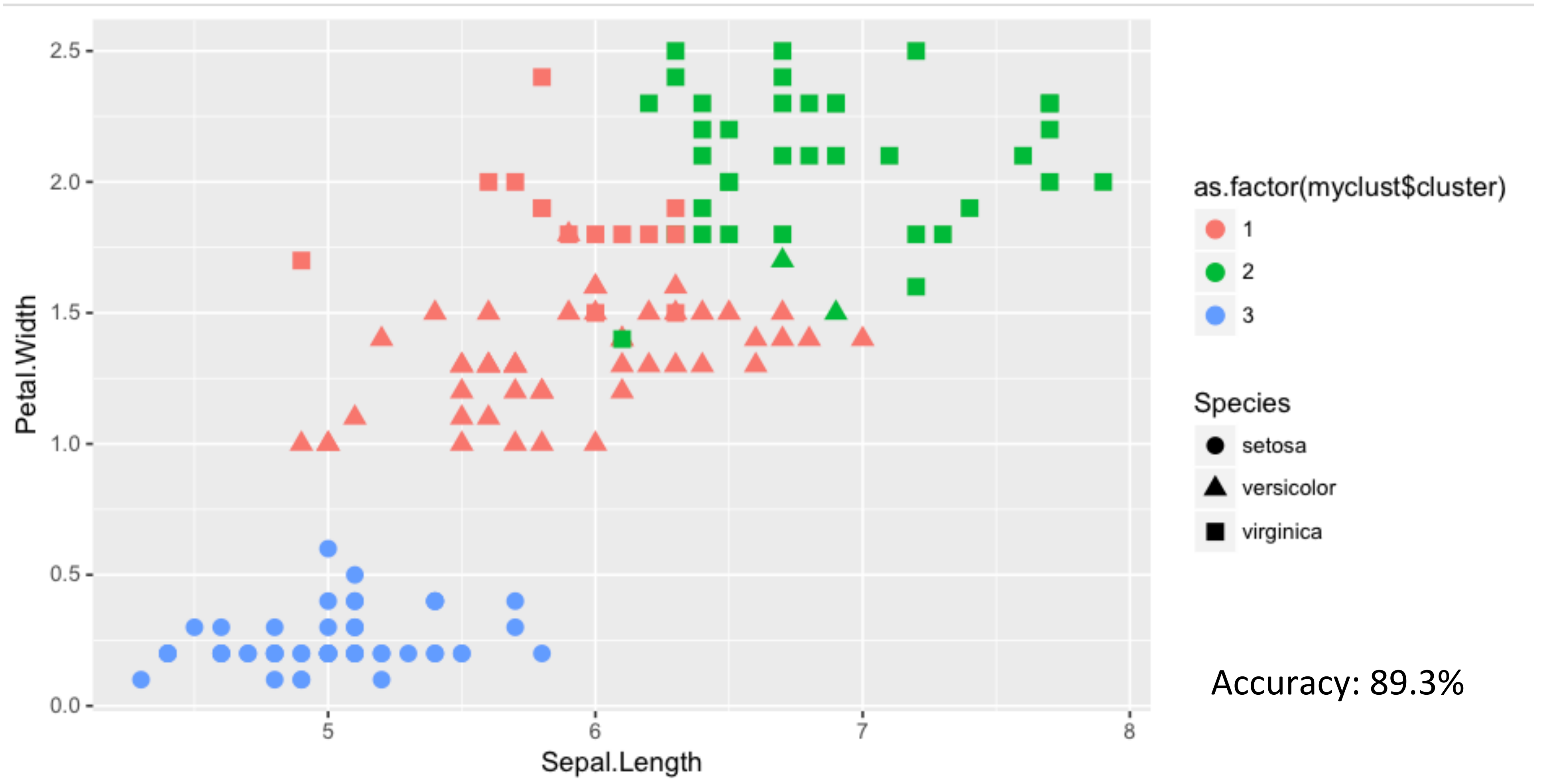
Choose k in advance, and then maximize the compactness of k clusters



(Image from http://simplystatistics.org/2014/02/18/k-means-clustering-in-a-gif/)

K-means clustering

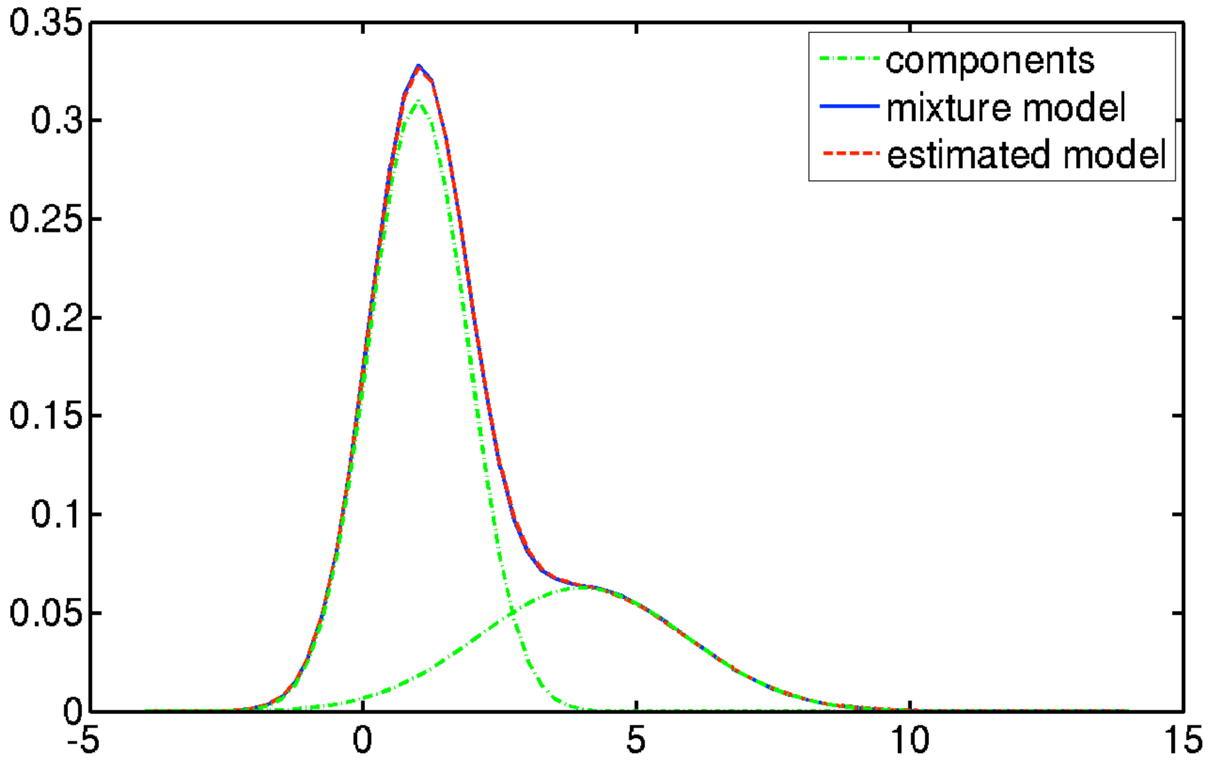
myclust=kmeans(iris[,1:4],3,iter.max=1000)



Accuracy: 89.3%

Gaussian mixture models

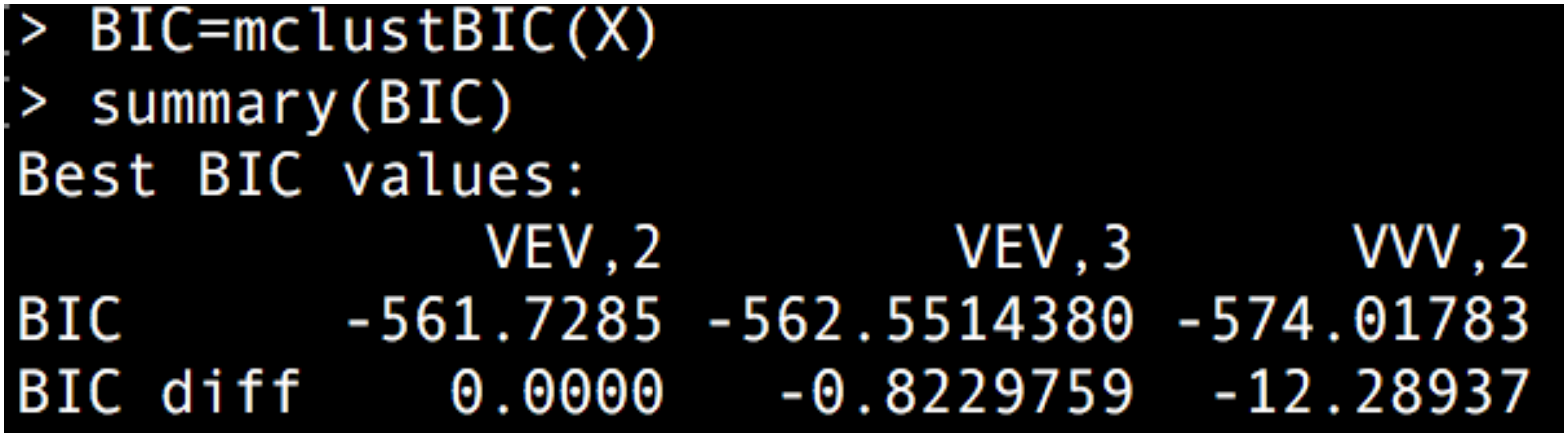
Model each cluster as a Gaussian with fitted characteristics



(Image from Mathworks.com)

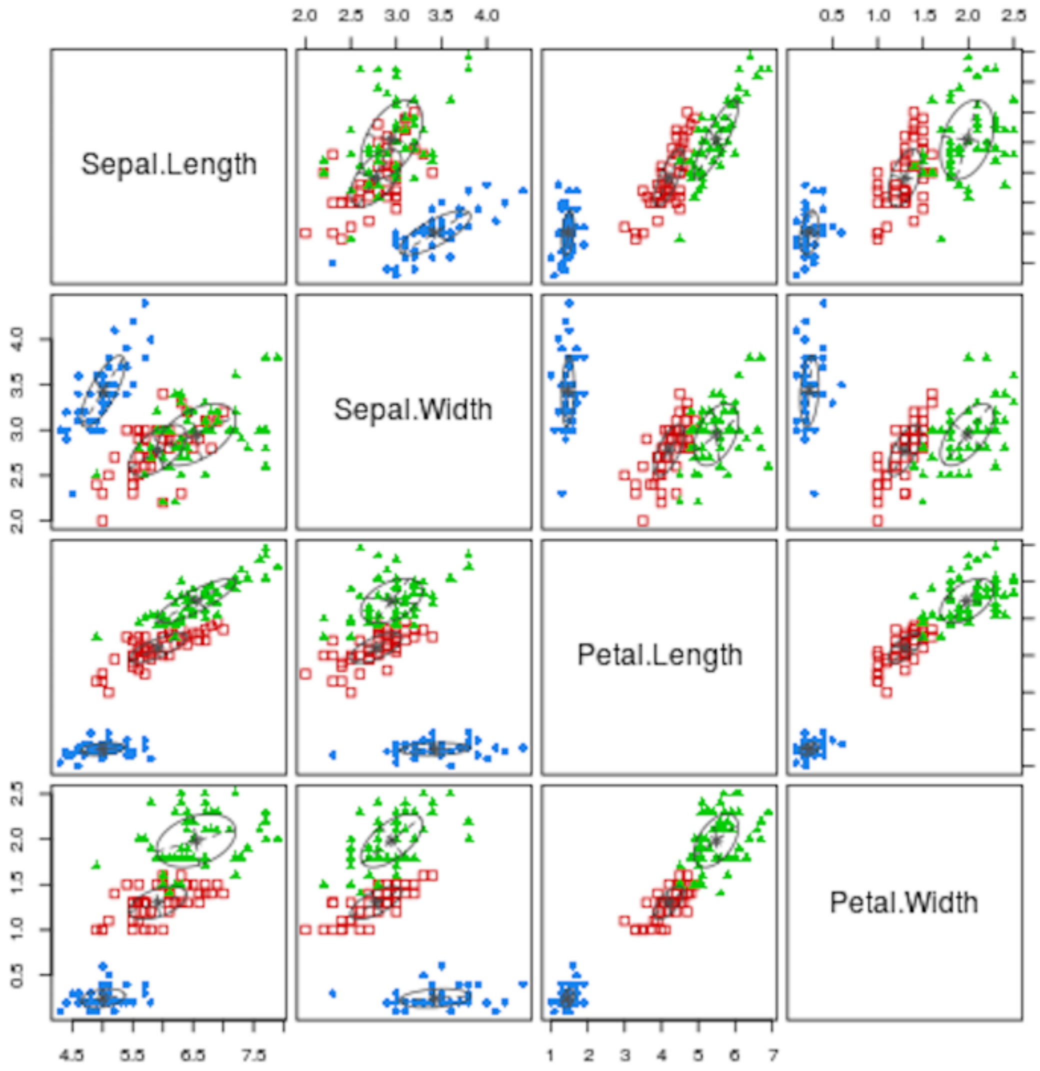
Gaussian mixture models

Model selection: Use Bayesian information criterion



```
> BIC=mclustBIC(X)
> summary(BIC)
Best BIC values:
               VEV,2        VEV,3       VVV,2
BIC        -561.7285 -562.5514380 -574.01783
BIC diff      0.0000   -0.8229759  -12.28937
```

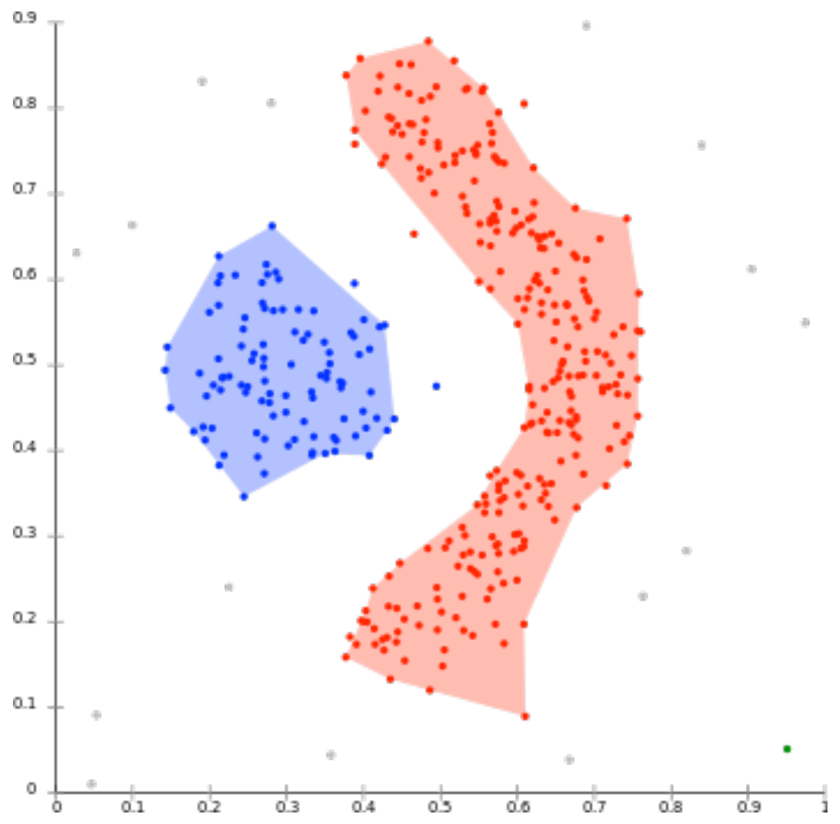
GMM on Iris data



97% accuracy
(If we get
3 models…)

DBSCAN for density-based clustering

Instead of looking for clumped data, find separations in low density regions



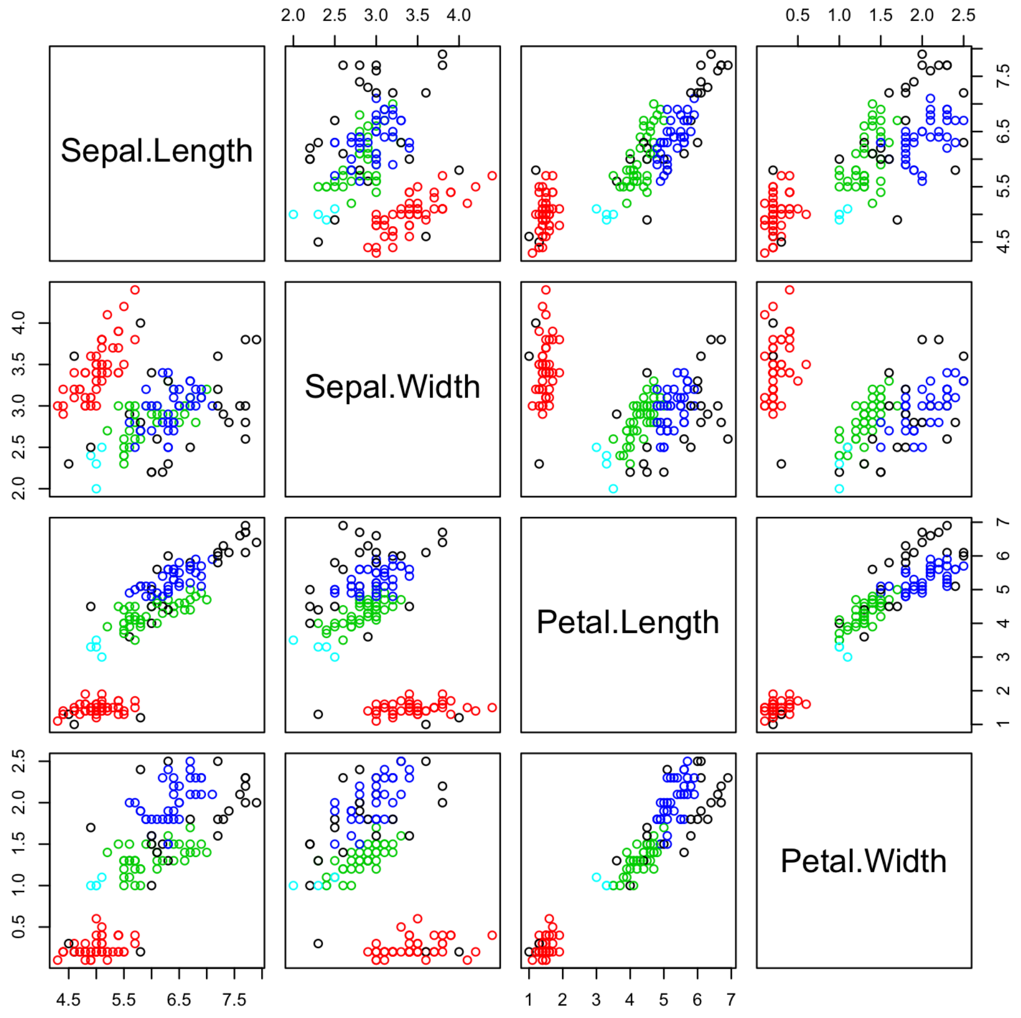
(Image from wikipedia user Chire)

DBSCAN for density-based clustering

Instead of looking for clumped data, find separations in low density regions

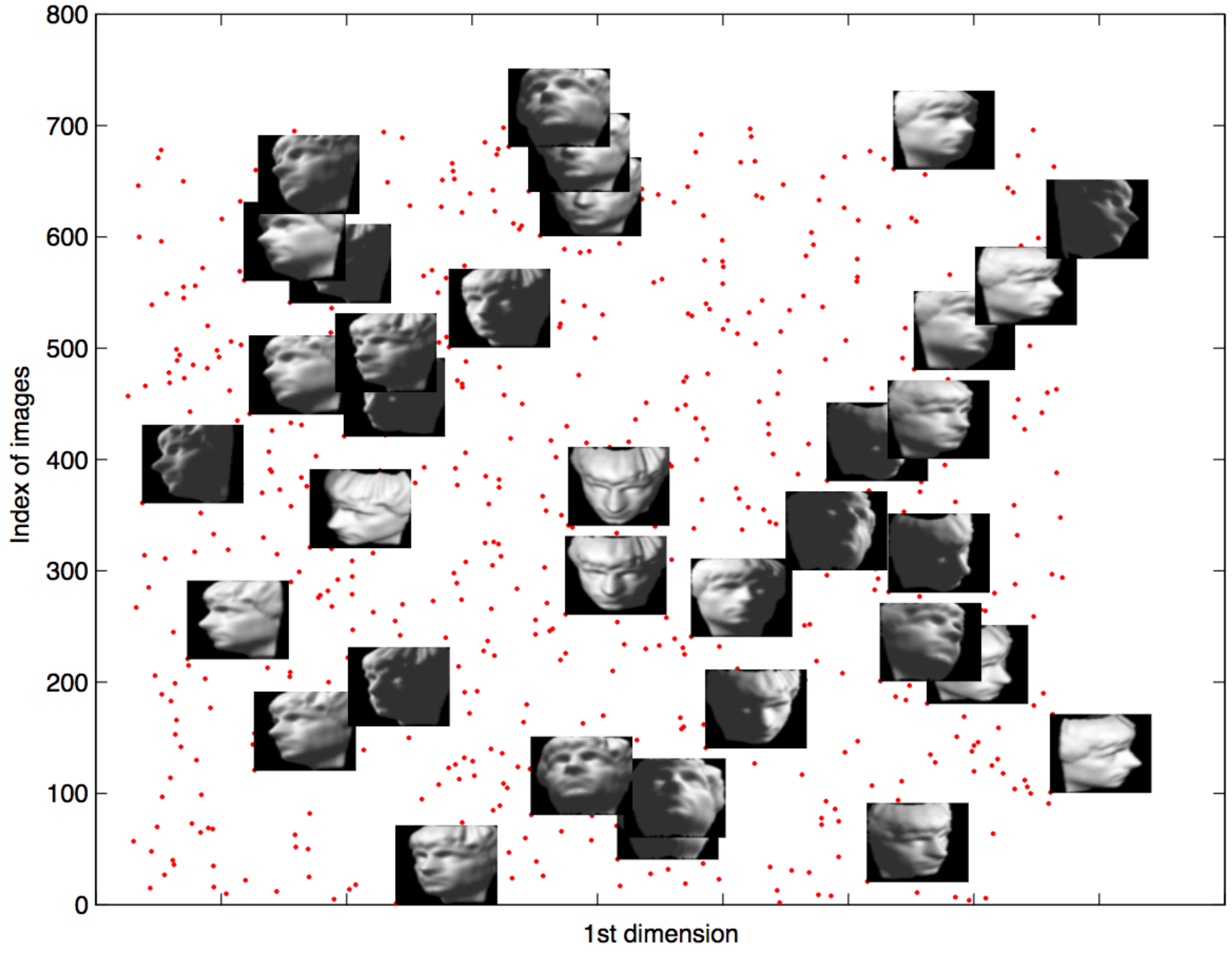
94% accuracy
(ignoring outliers)

79% accuracy
(including outliers)



db <- dbscan(x, eps = .4, minPts = 4)

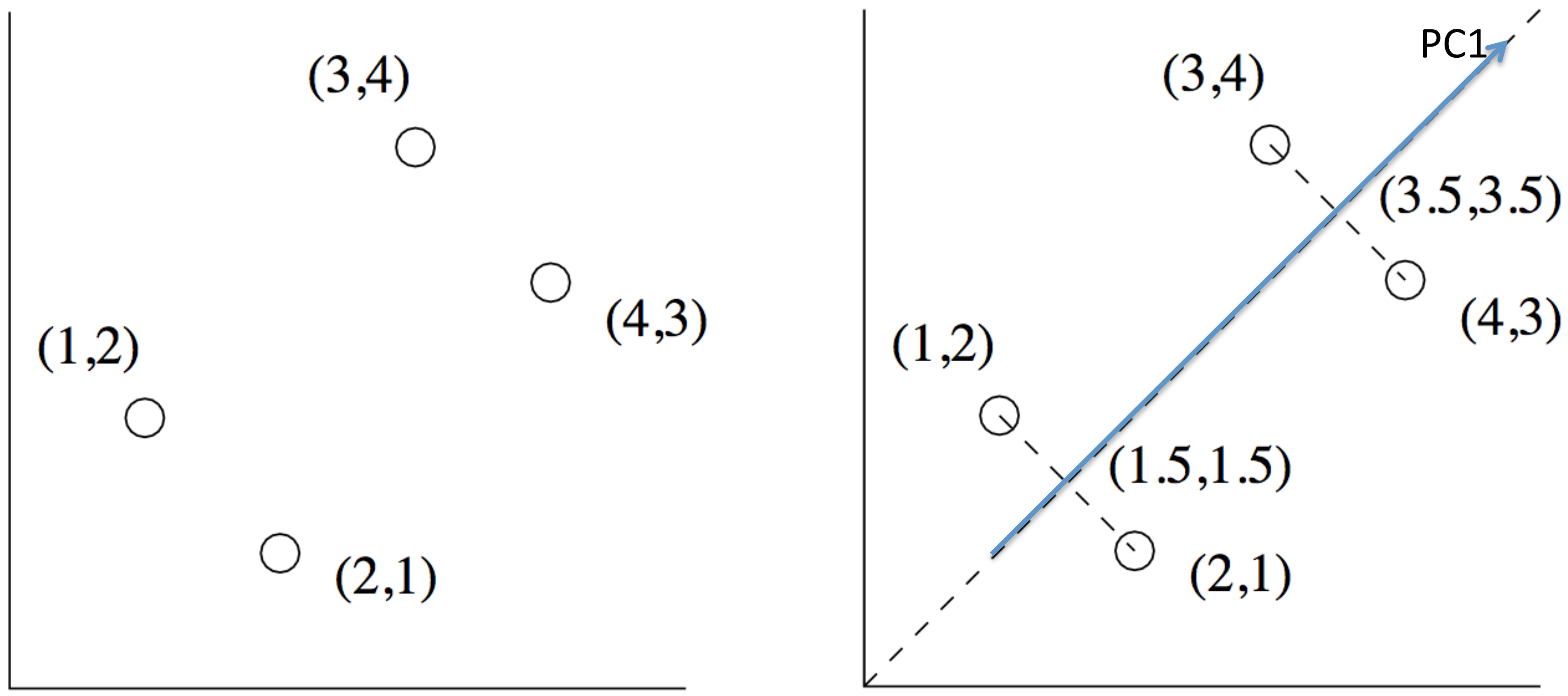
Dimensionality reduction and visualization



(Image from Ali Ghodsi, U. Waterloo)

Principal component analysis

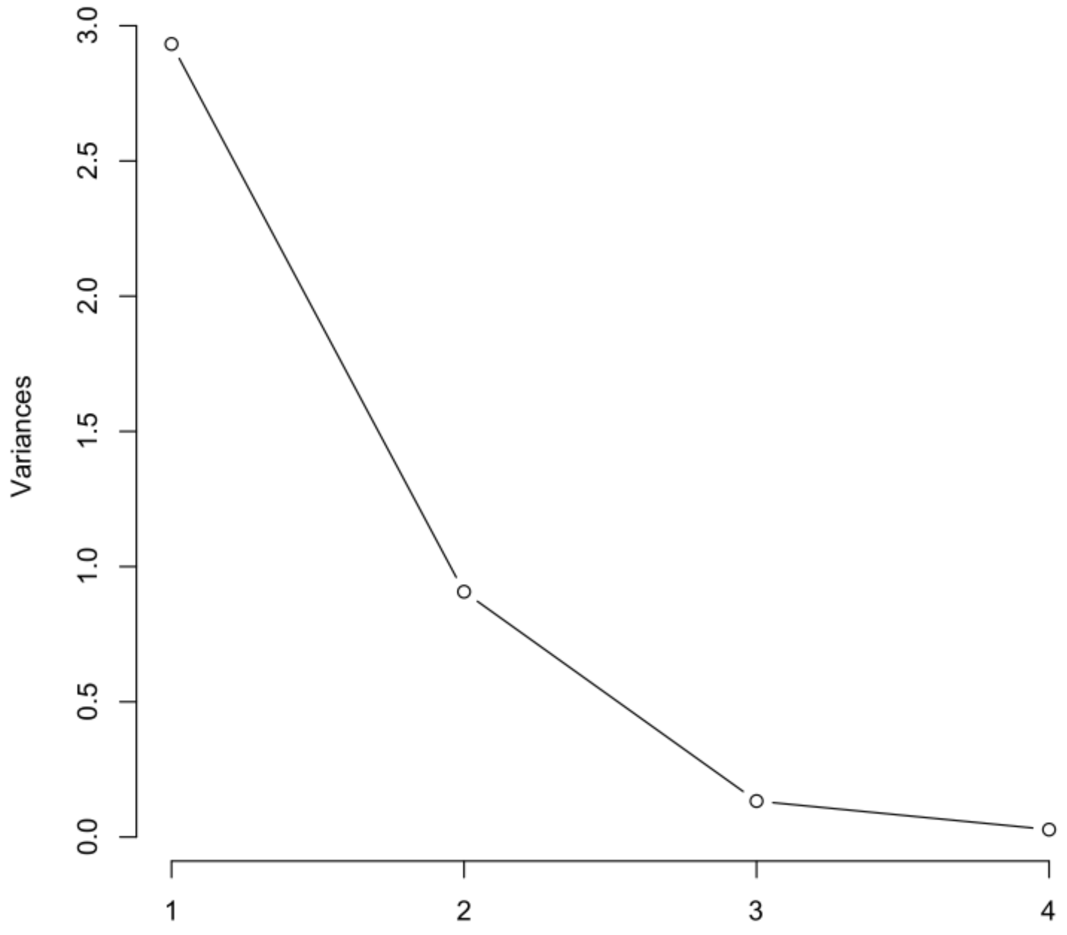
Find linear combinations of original data along which the variance is maximized



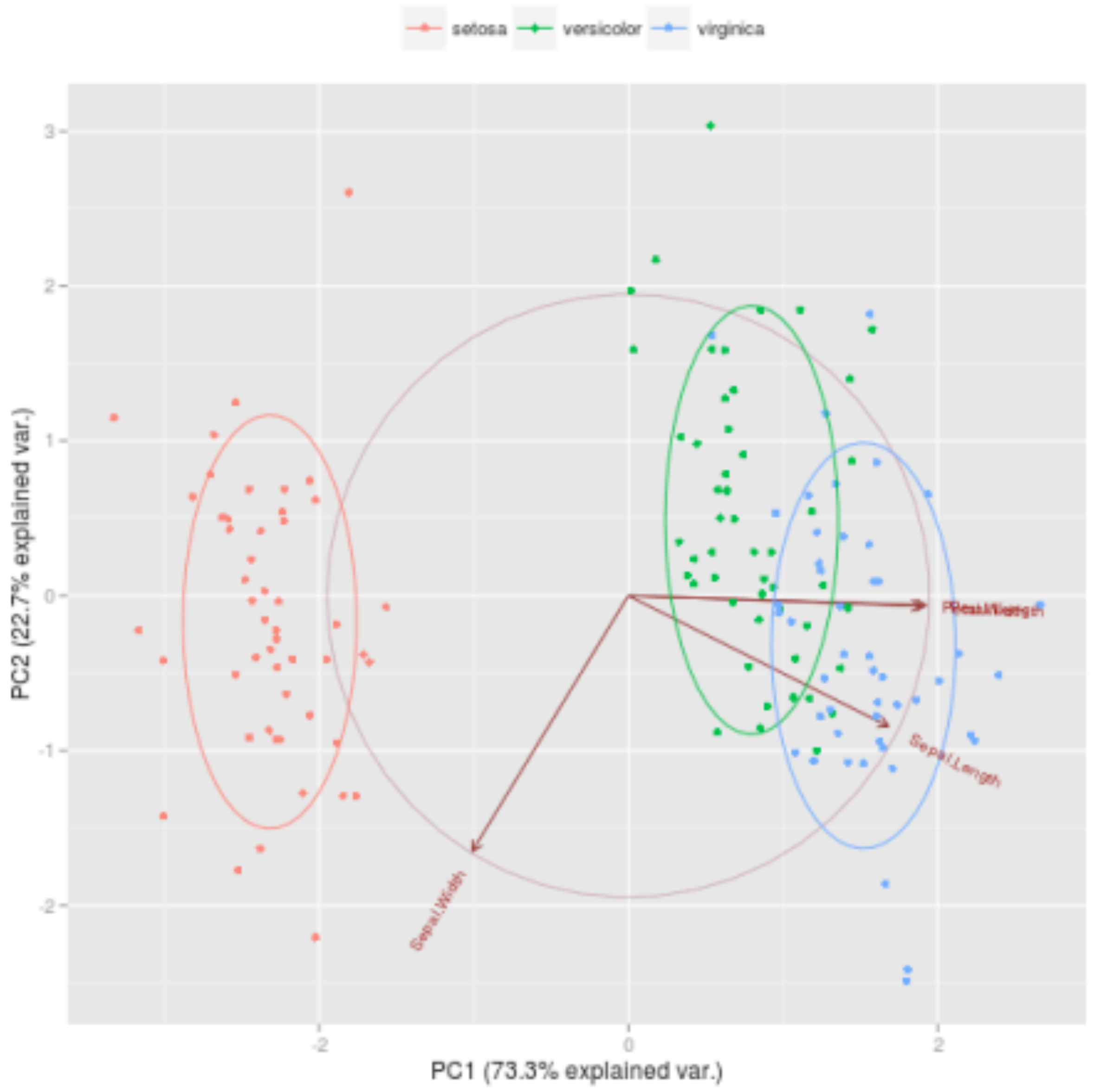
(Adapted from Jeff Ullman's *Mining Massive Datasets*)

Principal component analysis

```
pca.ir = prcomp(log.ir, center=TRUE, scale.=TRUE)
plot(pca.ir, type="l")
```



Principal component analysis



(Image from Thiago Martins)

Nonlinear methods for dimensionality reduction

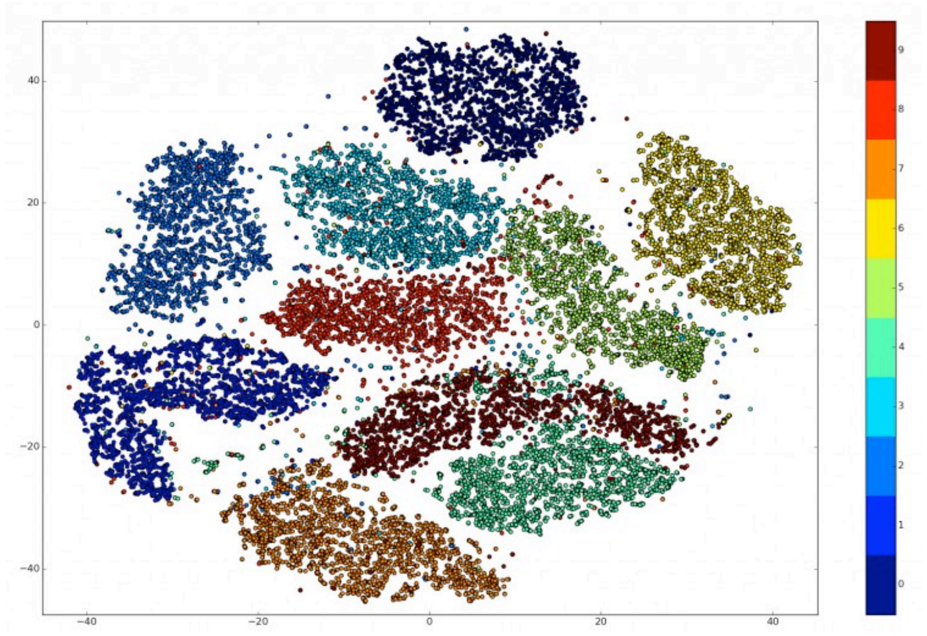
Nonlinear methods for dimensionality reduction

Example: t-SNE (t-distributed stochastic neighbor embedding)
Try to find reduced dimensions that reproduce locality of neighbors in high dimensions as well as possible

Nonlinear methods for dimensionality reduction

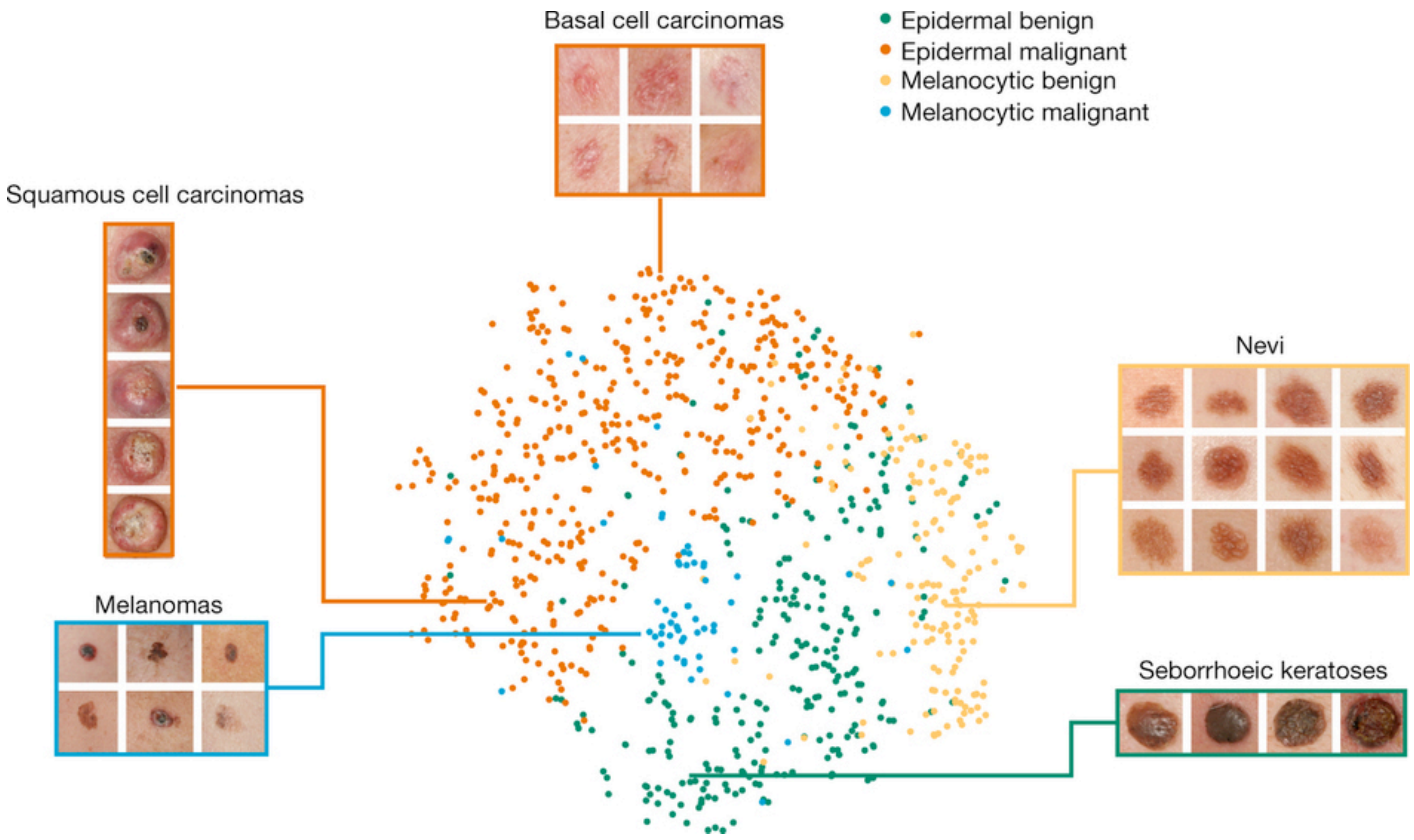
Example: t-SNE (t-distributed stochastic neighbor embedding)

Try to find reduced dimensions that reproduce locality of neighbors in high dimensions as well as possible



(Left image from MNIST dataset, right from https://indico.io/blog/visualizing-with-t-sne/)

Nonlinear methods for dimensionality reduction



Nature 542, 115–118 (02 February 2017)

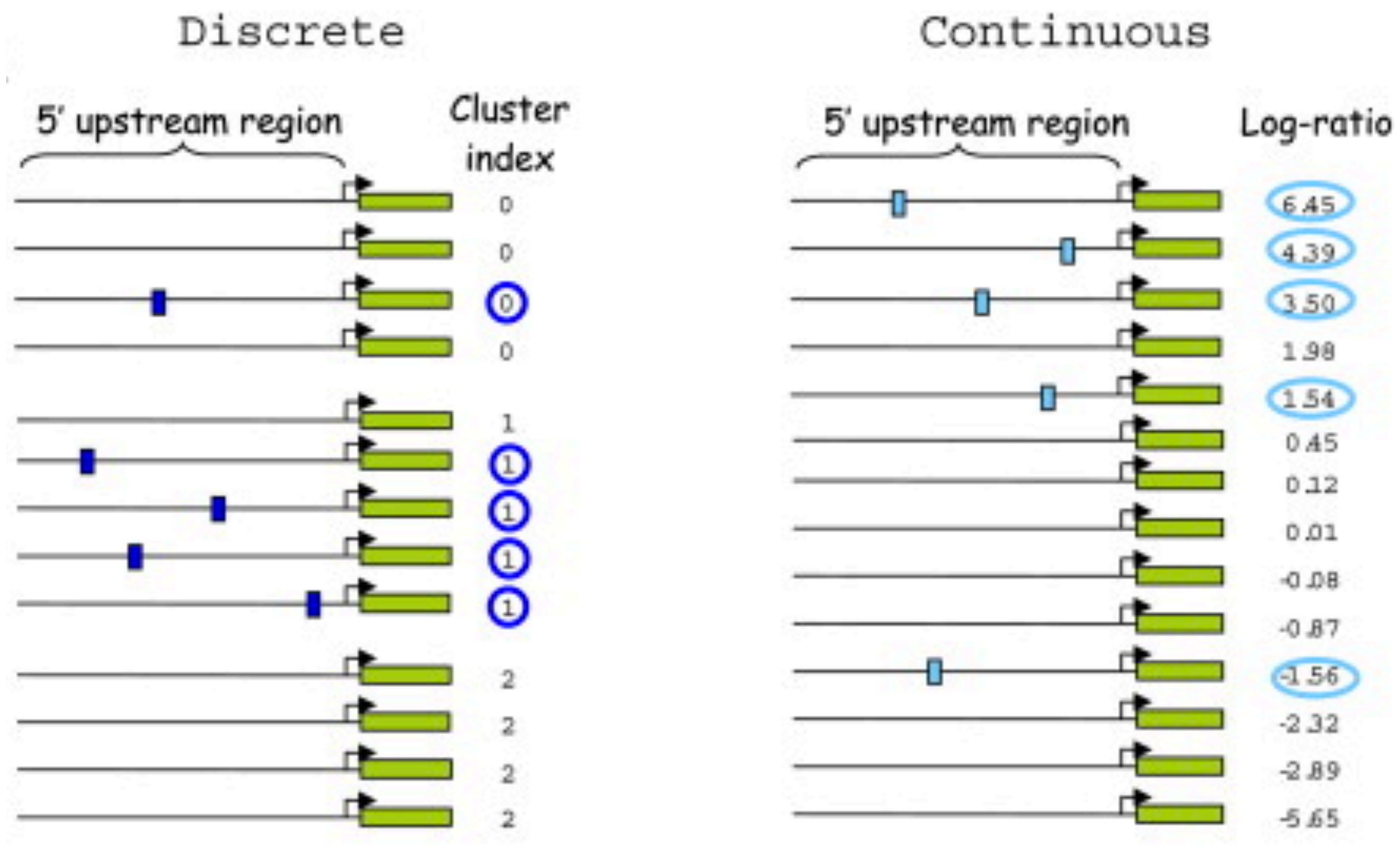
Feature selection

- Focus on removal of features that give little information
- Some combination of art and a wide variety of automated methods
- More important for some machine learning methods than others
- Need to have as much information as possible when determining features (if you have any choice)

Outline

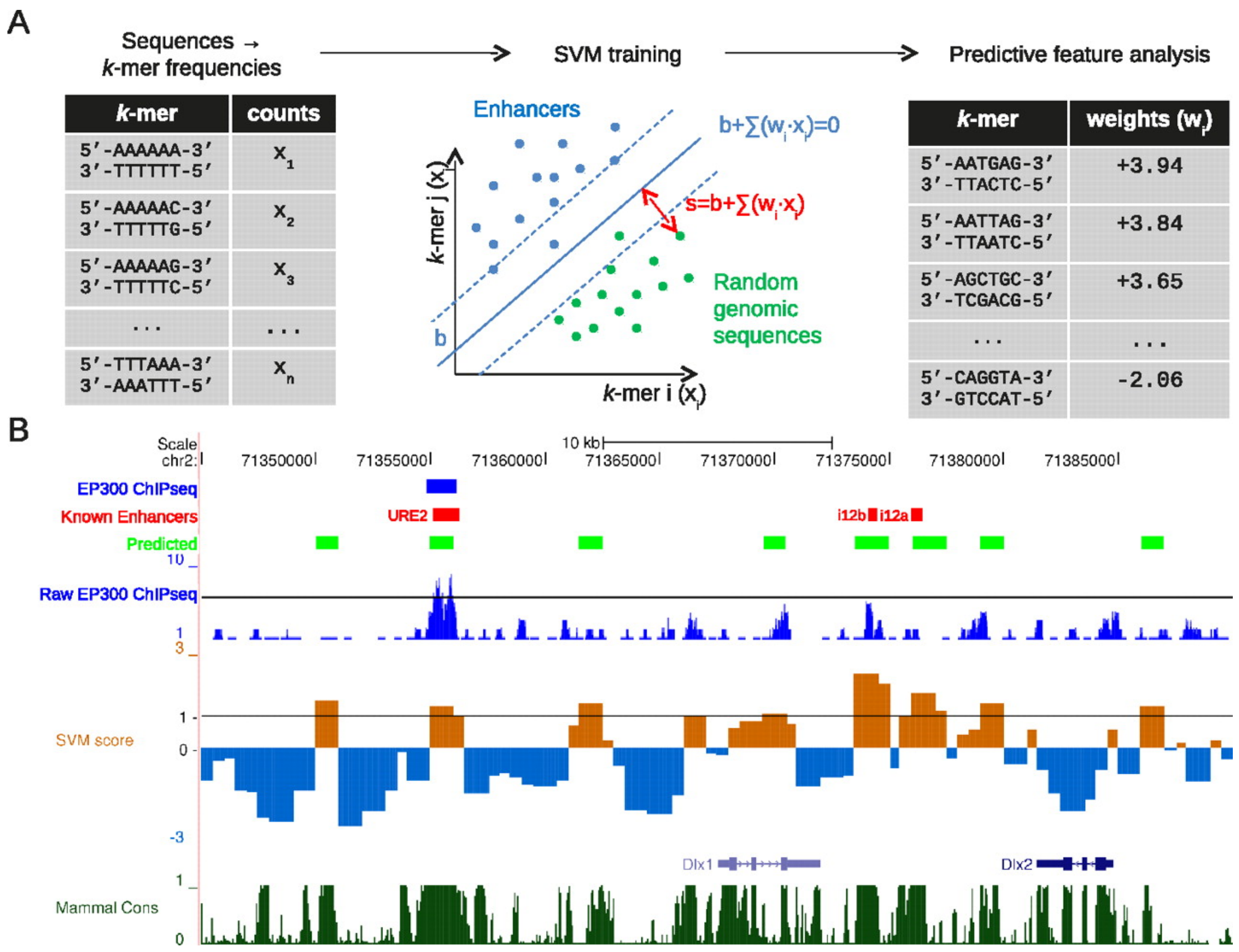
- Data import and management in R
- Overview of machine learning
- Common machine learning methods
- **Applications of machine learning in biology**

Motif identification!



(Image from Elemento *et al.*, Mol. Cell 2007)

Motif identification!

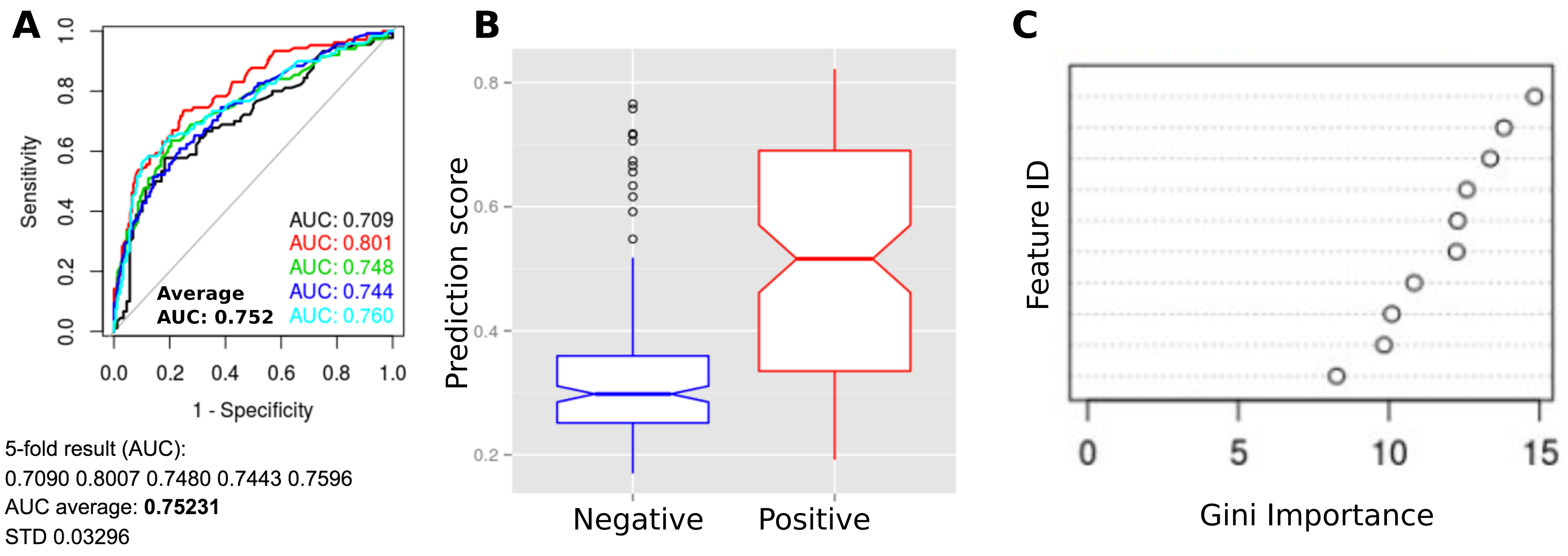


k-mer	counts
5′-AAAAAA-3′ 3′-TTTTTT-5′	x_1
5′-AAAAAC-3′ 3′-TTTTTG-5′	x_2
5′-AAAAAG-3′ 3′-TTTTTC-5′	x_3
...	...
5′-TTTAAA-3′ 3′-AAATTT-5′	x_n

k-mer	weights (w_i)
5′-AATGAG-3′ 3′-TTACTC-5′	+3.94
5′-AATTAG-3′ 3′-TTAATC-5′	+3.84
5′-AGCTGC-3′ 3′-TCGACG-5′	+3.65
...	...
5′-CAGGTA-3′ 3′-GTCCAT-5′	-2.06

Dongwon Lee et al. Genome Res. 2011;21:2167-2180

Functional sequence analysis

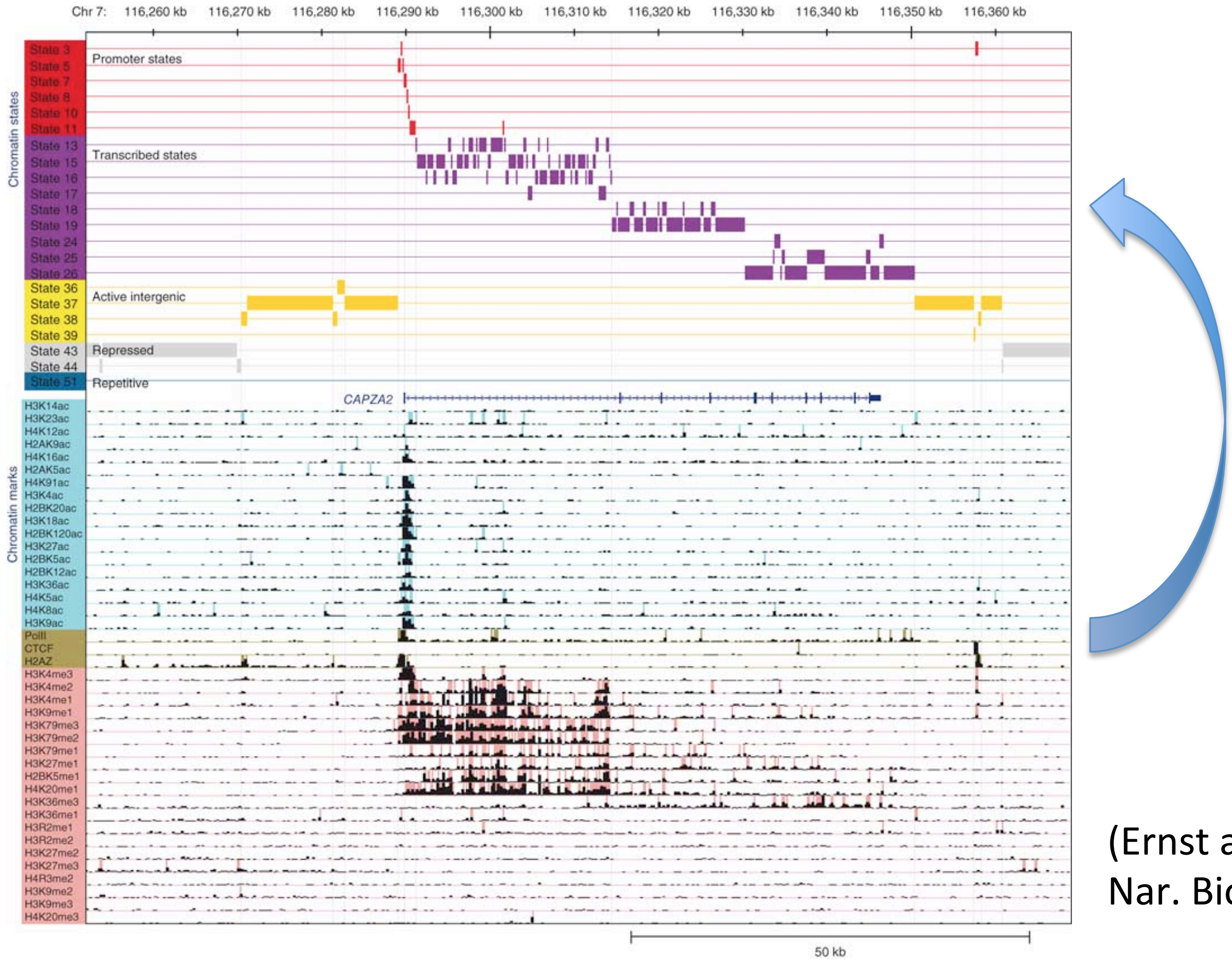


5-fold result (AUC):
0.7090 0.8007 0.7480 0.7443 0.7596
AUC average: **0.75231**
STD 0.03296

Identify key features of rho-dependent terminators: specific motifs, sequence composition, RNA folding energy

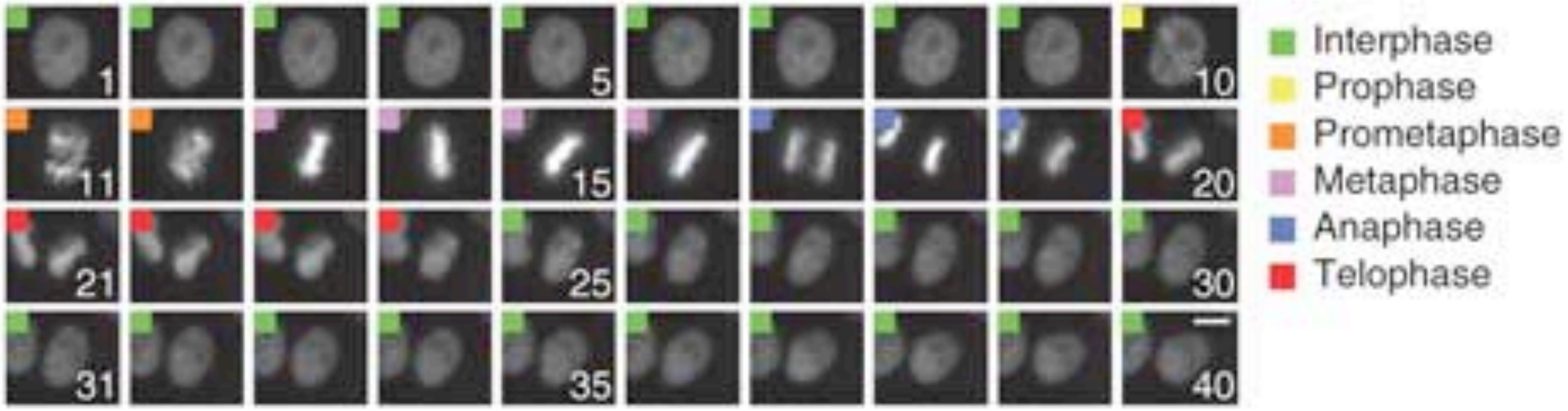
(Image from Taeho Jo)

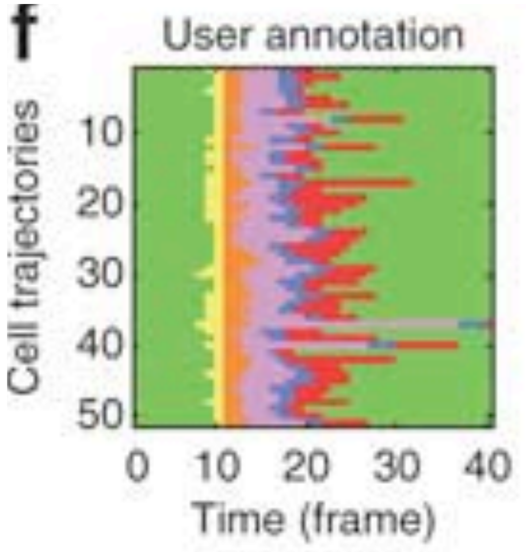
Identifying chromatin states from ENCODE data

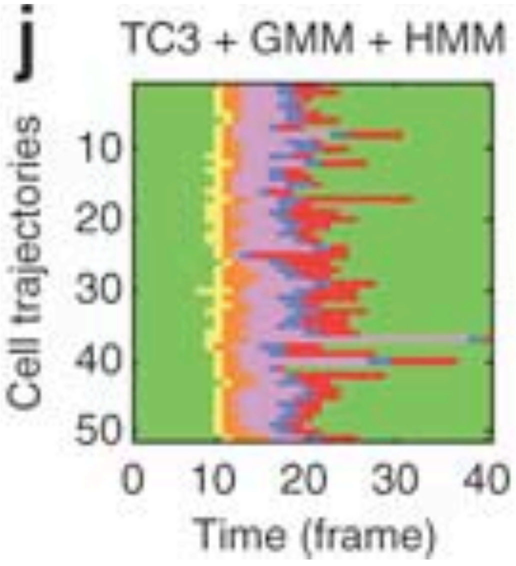


(Ernst and Kellis; Nar. Biotech. 2010)

Analysis of microscopy data

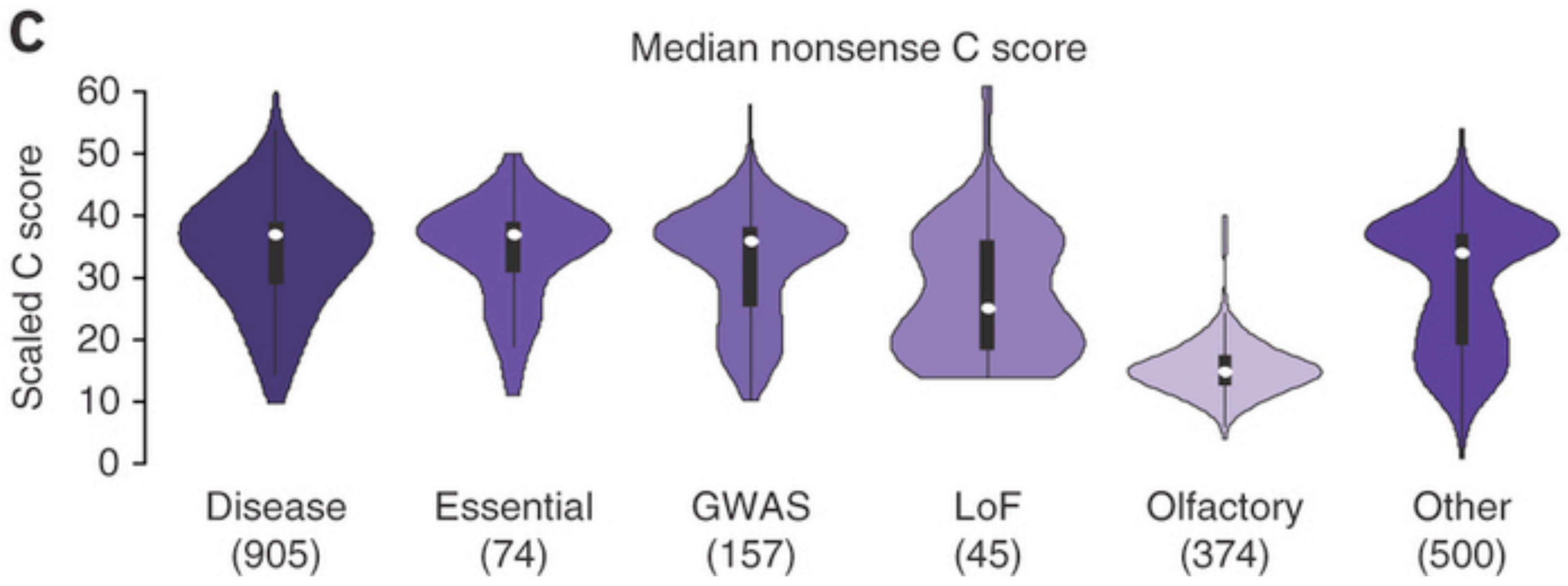






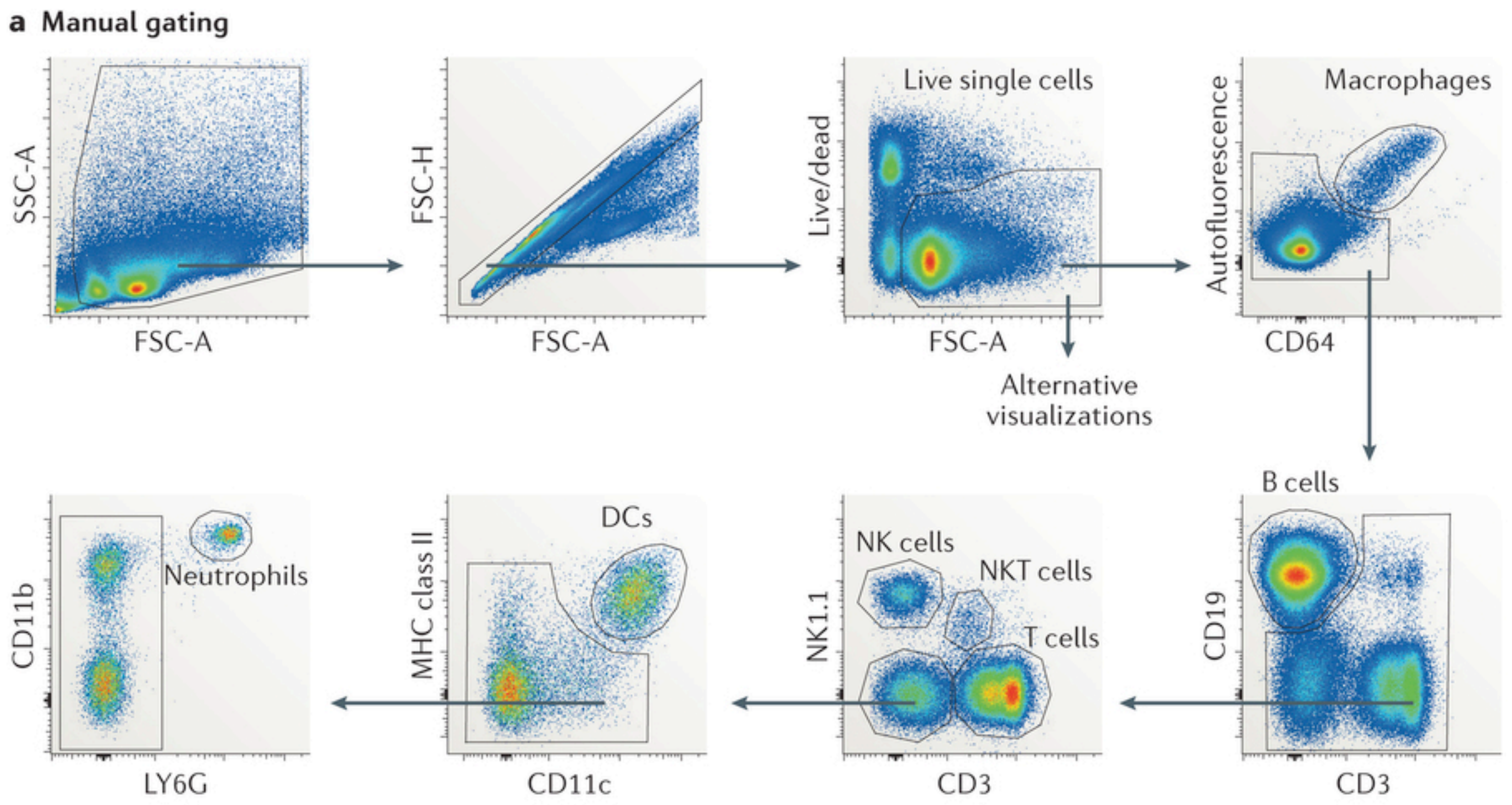
(Zhong et al.,
Nat. Meth. 2012)

Prediction of disease-causing mutations



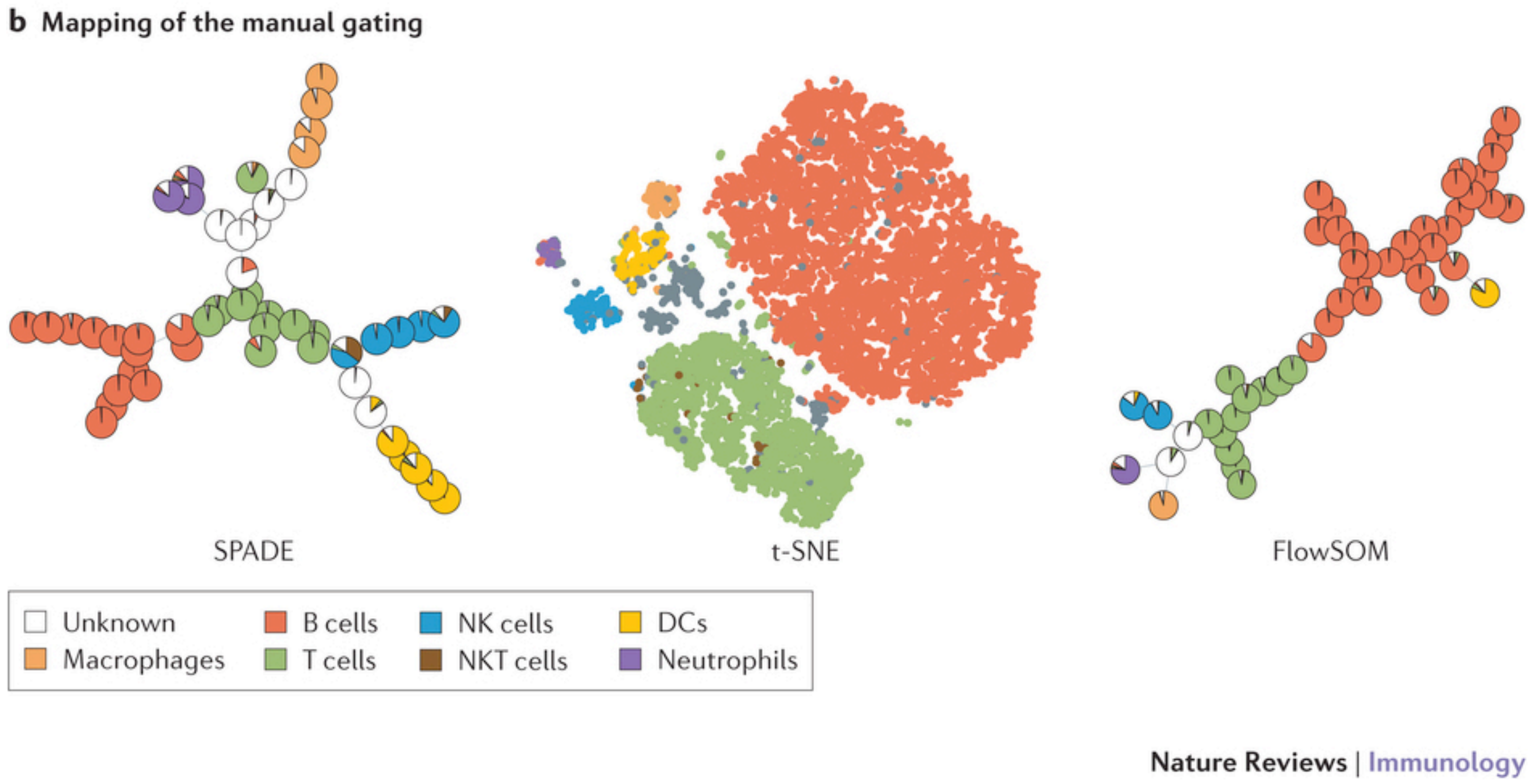
(Kircher et al., Nat. Genet. 20104)

Analysis of flow cytometry data



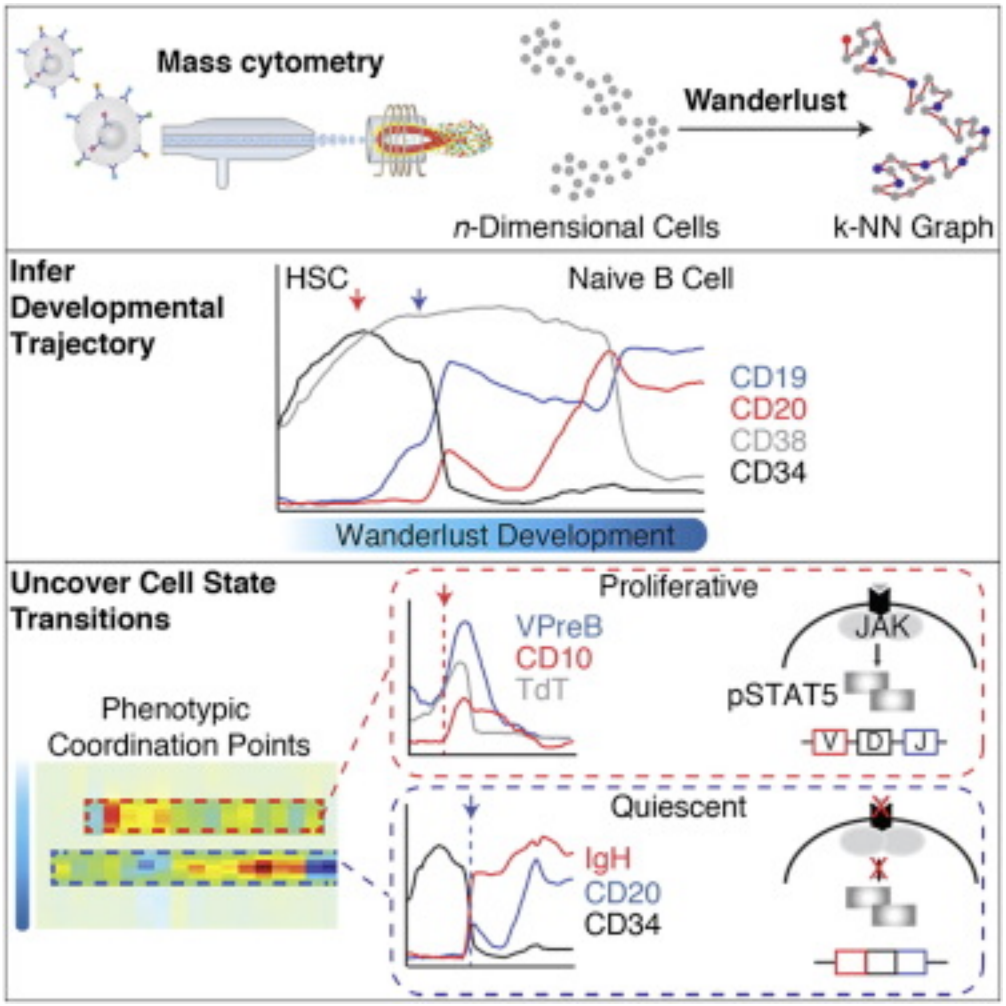
(Saeys et al., Nat. Rev. Immunol. 2016)

Analysis of flow cytometry data



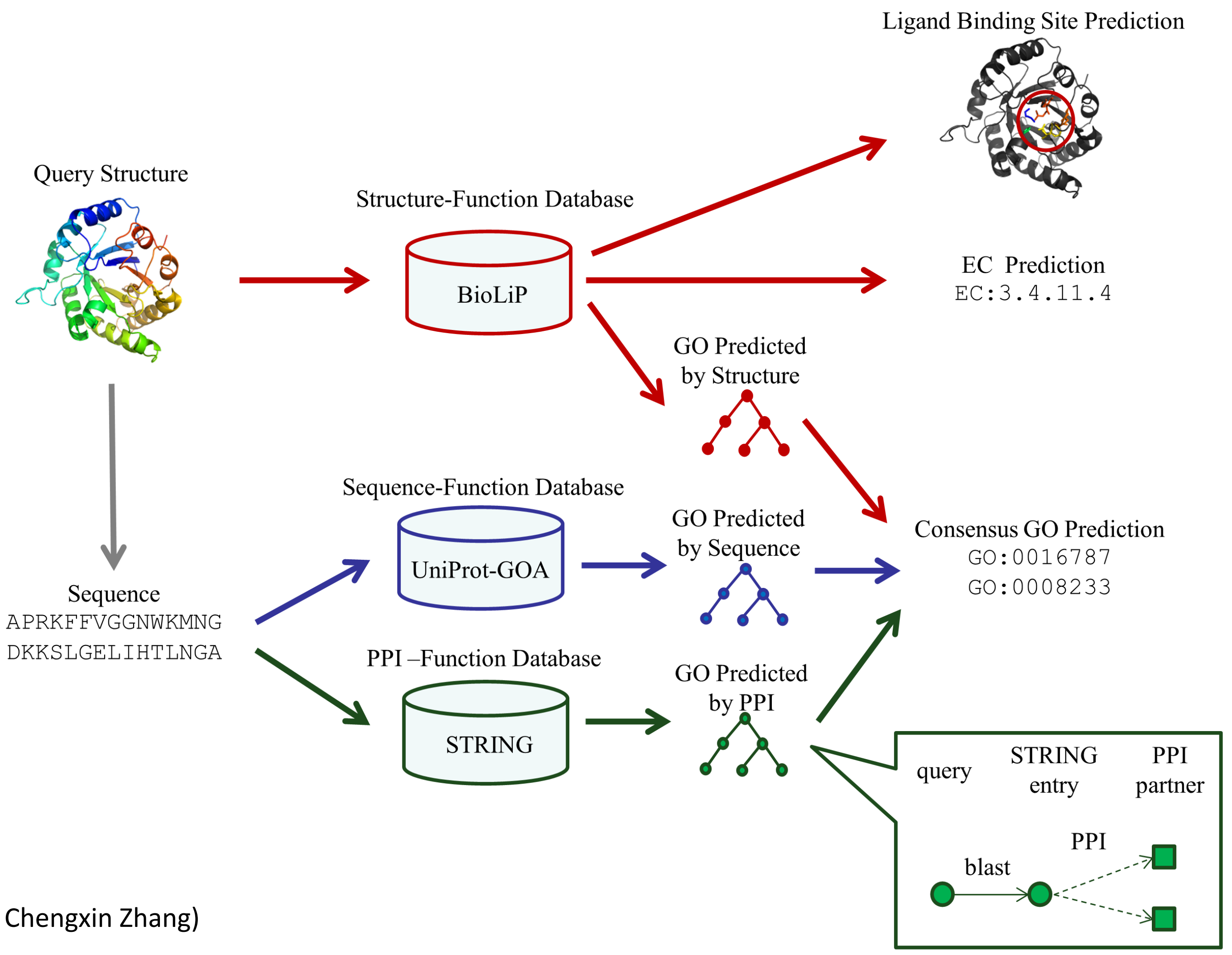
(Saeys et al., Nat. Rev. Immunol. 2016)

Automated inference of developmental pathways



(Bendall et al., Cell 2014)

Predicting the functions of un-annotated genes



(Image from Chengxin Zhang)