



# Some packages relative to machine learning in R

---

林松江

2005/9/30



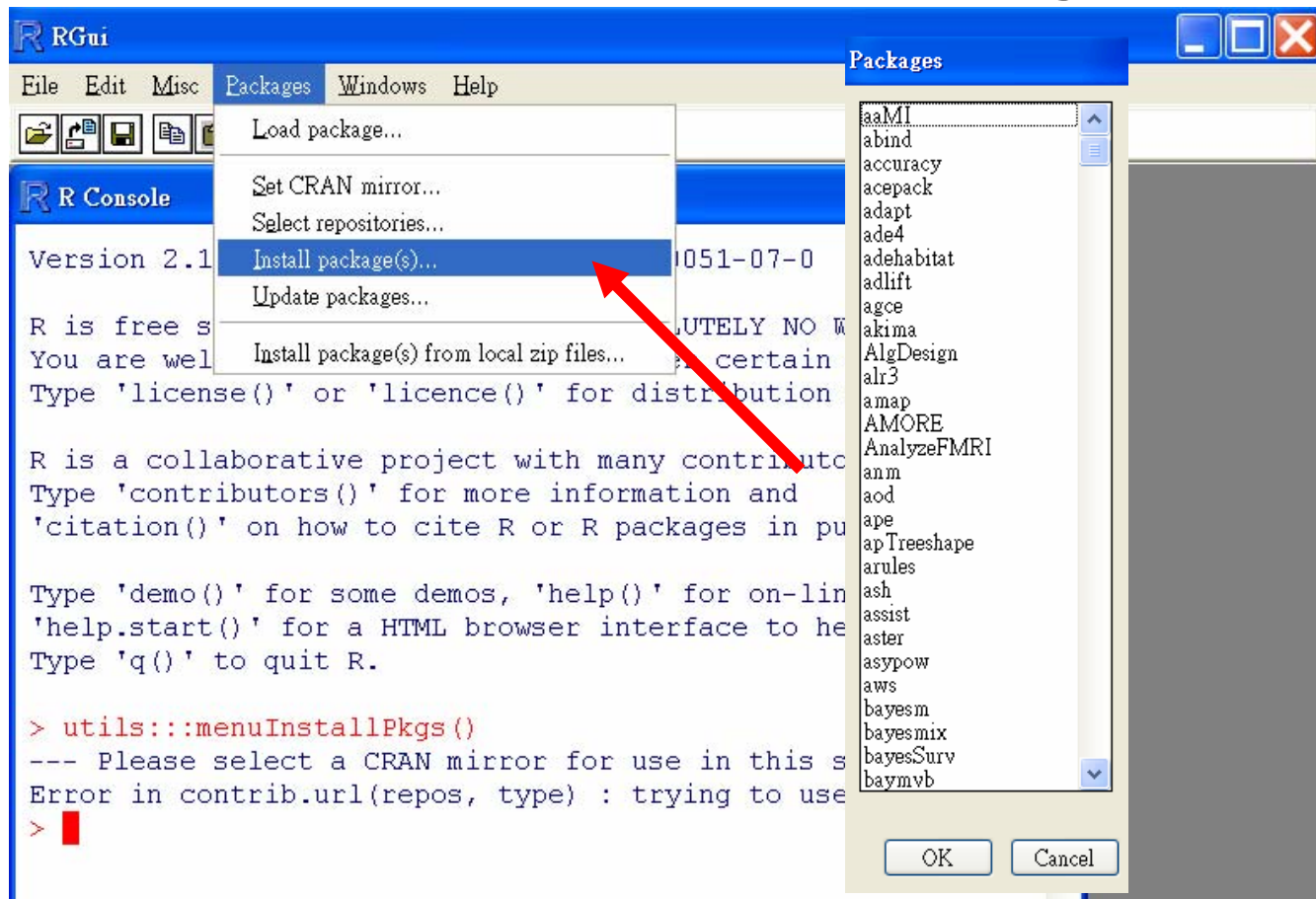
# Machine learning methods

---

- **Supervised Learning**, where we get a set of training inputs and outputs
  - classification, regression
    - Tree, SVM, KNN, LDA, ...
- **Unsupervised Learning**, where we are interested in capturing inherent organization in the data
  - clustering, density estimation
    - K-mean, EM, SOM, ...

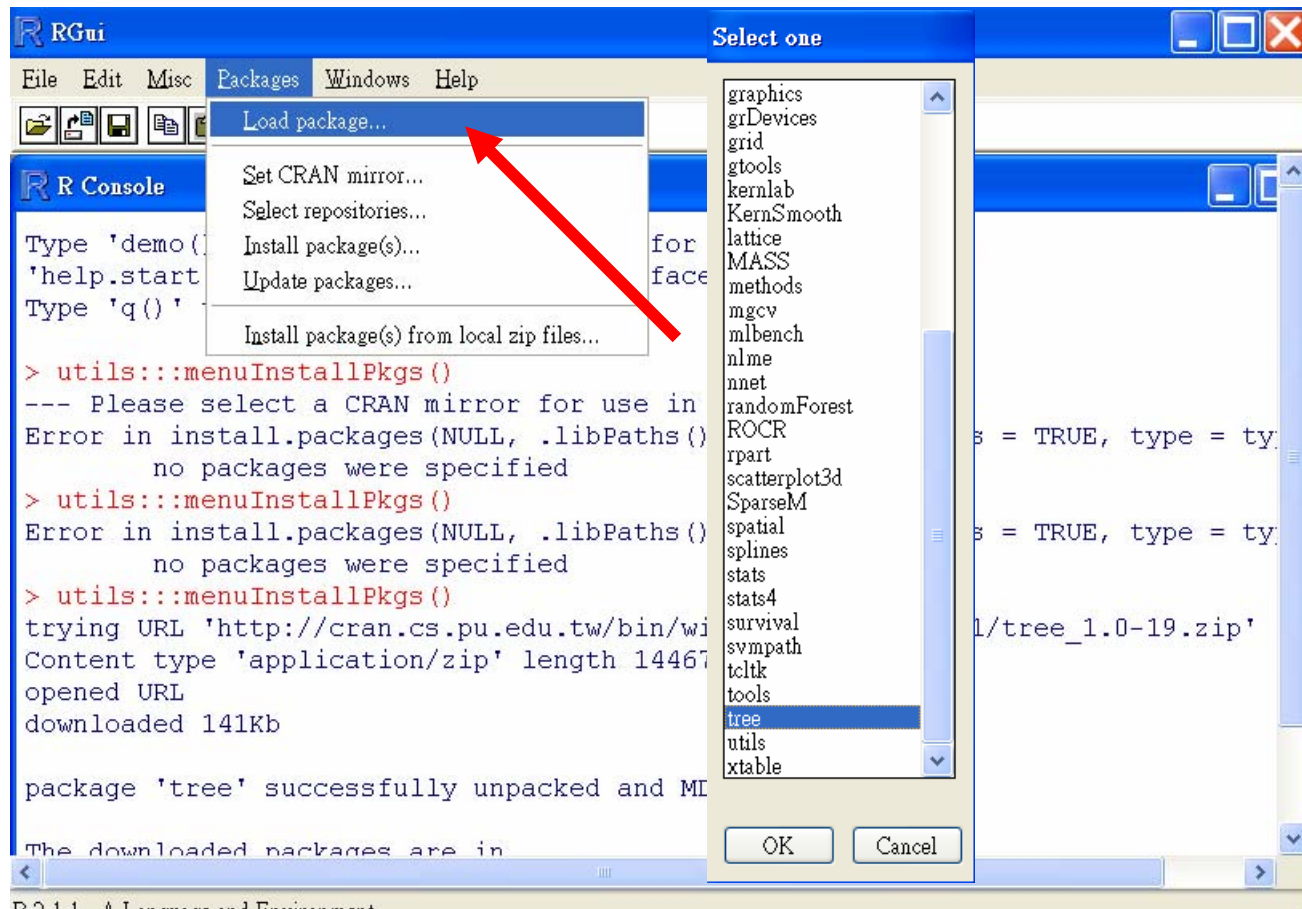
# Install new packages “tree”

- For example: `> install.packages(“tree”)`



# Load packages “tree”

- For example: `> library(tree)`



The screenshot shows the RGui interface. The 'Packages' menu is open, and the 'Load package...' option is highlighted with a red arrow. Below the menu, the R Console shows the following output:

```
> utils:::menuInstallPkgs ()  
--- Please select a CRAN mirror for use in  
Error in install.packages (NULL, .libPaths ())  
  no packages were specified  
> utils:::menuInstallPkgs ()  
Error in install.packages (NULL, .libPaths ())  
  no packages were specified  
> utils:::menuInstallPkgs ()  
trying URL 'http://cran.cs.pu.edu.tw/bin/wi  
Content type 'application/zip' length 14467  
opened URL  
downloaded 141Kb  
  
package 'tree' successfully unpacked and MI  
The downloaded packages are in
```

The 'Select one' dialog box is open, showing a list of packages. The 'tree' package is selected and highlighted in blue. The list includes:

- graphics
- grDevices
- grid
- gtools
- kernlab
- KernSmooth
- lattice
- MASS
- methods
- mgcv
- mlbench
- nlme
- nnet
- randomForest
- ROCR
- rpart
- scatterplot3d
- SparseM
- spatial
- splines
- stats
- stats4
- survival
- svmpath
- tcltk
- tools
- tree
- utils
- xtable

The 'OK' button is visible at the bottom of the dialog box.

# Help for new packages

The screenshot shows the RGui interface with the Help menu open. The menu items are: Console, EAQ on R, FAQ on R for Windows, Manuals (in PDF), R functions (text)..., **Html help** (highlighted with a red arrow), Search help..., search.r-project.org..., Apropos..., R Project home page, CRAN home page, and About. The R Console shows the following output:

```
Error in install.packages
no packages were
> utils:::menuInstallPkgs
Error in install.packages
no packages were
> utils:::menuInstallPkgs
trying URL 'http://cran.c
Content type 'application
opened URL
downloaded 141Kb

package 'tree' successful

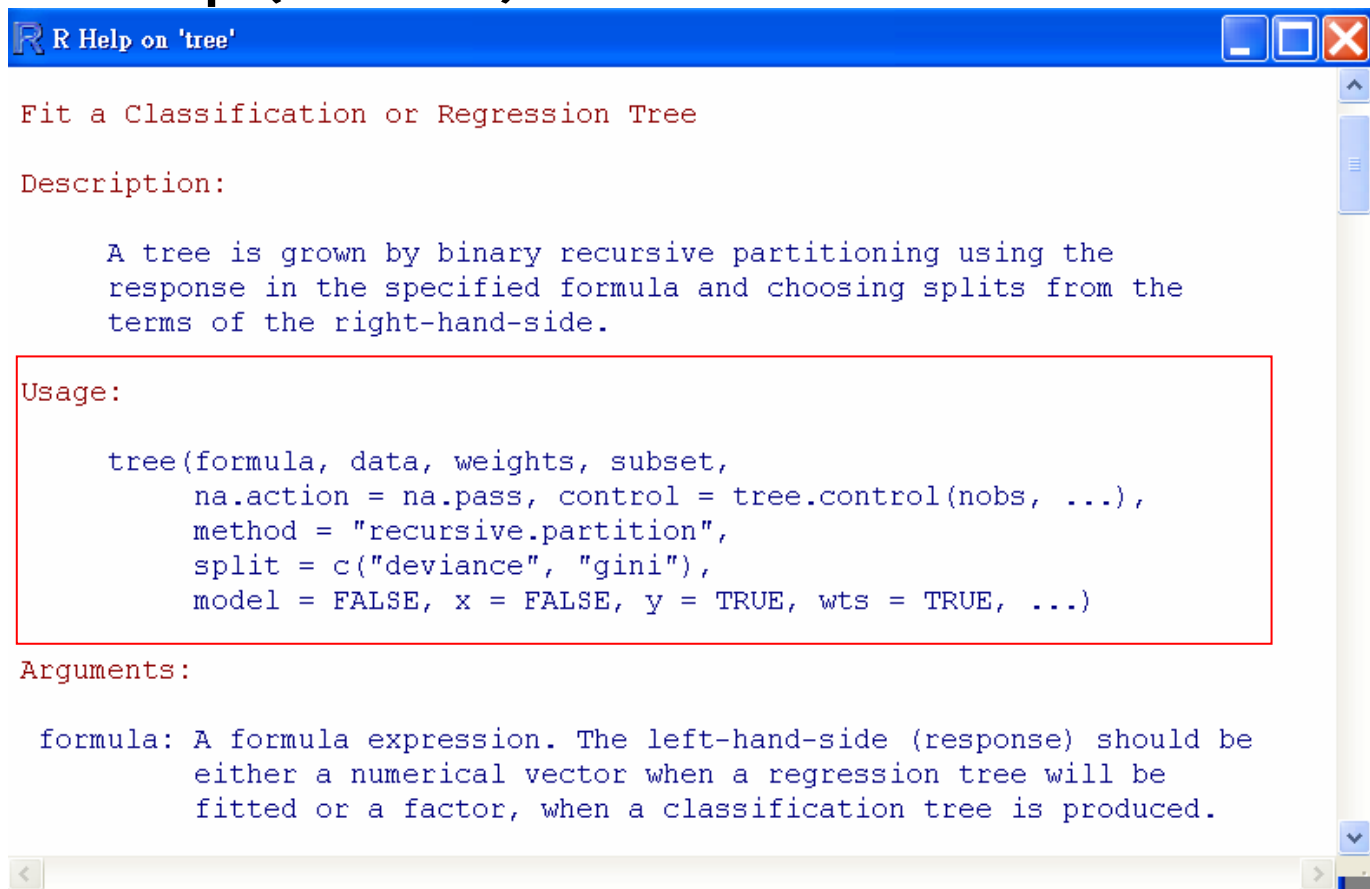
The downloaded packages are in
C:\Documents and Settings\Administrator\L
updating HTML package descriptions
> local({pkg <- select.list(sort(.packages(all.available = TRUE)))
+ if(nchar(pkg)) library(pkg, character.only=TRUE)})
> library(tree)
> local({pkg <- select.list(sort(.packages(all.available = TRUE)))
```

The web browser window displays the R Statistical Data Analysis website. The page title is "Statistical Data Analysis" and features the R logo. Below the logo, there is a "Manuals" section with links to "An Introduction to R", "Writing R Extensions", "The R Language Definition", "R Data Import/Export", and "R Installation and Administration". There is also a "Reference" section and a "Search Engine & Keywords" section.

R 2.1.1 - A Language and Environment

# Usage for tree: input argument

- `> help("tree")`

A screenshot of the R Help window for the 'tree' package. The window title is 'R Help on 'tree''. The content is as follows:

```
Fit a Classification or Regression Tree

Description:

A tree is grown by binary recursive partitioning using the
response in the specified formula and choosing splits from the
terms of the right-hand-side.

Usage:

tree(formula, data, weights, subset,
      na.action = na.pass, control = tree.control(nobs, ...),
      method = "recursive.partition",
      split = c("deviance", "gini"),
      model = FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)

Arguments:

formula: A formula expression. The left-hand-side (response) should be
either a numerical vector when a regression tree will be
fitted or a factor, when a classification tree is produced.
```

# Usage for tree: Examples

R Help on 'tree'

Cambridge University Press, Cambridge. Chapter 7.

See Also:

'tree.control', 'prune.tree', 'predict.tree', 'snip.tree'

Examples:

```
library(MASS)
data(cpus)
cpus.ltr <- tree(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus)
cpus.ltr
summary(cpus.ltr)
plot(cpus.ltr); text(cpus.ltr)

data(iris)
ir.tr <- tree(Species ~., iris)
ir.tr
summary(ir.tr)
```

# Dataset in R

■ > data()

```
R data sets
Data sets in package 'datasets':

AirPassengers      Monthly Airline Passenger Numbers 1949-1960
BJsales            Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD                Biochemical Oxygen Demand
CO2                Carbon Dioxide uptake in grass plants
ChickWeight        Weight versus age of chicks on different diets
DNase              Elisa assay of DNase
EuStockMarkets     Daily Closing Prices of Major European Stock
                  Indices, 1991-1998
Formaldehyde       Determination of Formaldehyde
HairEyeColor       Hair and Eye Color of Statistics Students
Harman23.cor        Harman Example 2.3
Harman74.cor        Harman Example 7.4
Indometh            Pharmacokinetics of Indomethicin
InsectSprays       Effectiveness of Insect Sprays
JohnsonJohnson    Quarterly Earnings per Johnson & Johnson Share
LakeHuron          Level of Lake Huron 1875-1972
LifeCycleSavings   Intercountry Life-Cycle Savings Data
Loblolly           Growth of Loblolly pine trees
Nile               Flow of the River Nile
Orange             Growth of orange trees
OrchardSprays      Potency of Orchard Sprays
```





# For example: load “iris” dataset

---

- `> data(iris)`

- `> ls()`

```
[1] "iris"
```

- `> iris[1:5,]`

	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




# Fit a Classification Tree

---

```
> ir.tr <- tree(Species ~., iris)
> ir.tr
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
 2) Petal.Length < 2.45 50 0.000 setosa ( 1.00000 0.00000 0.00000 ) *
 3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
   6) Petal.Width < 1.75 54 33.320 versicolor ( 0.00000 0.90741 0.09259 )
     12) Petal.Length < 4.95 48 9.721 versicolor ( 0.00000 0.97917 0.02083 )
       24) Sepal.Length < 5.15 5 5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
       25) Sepal.Length > 5.15 43 0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
     13) Petal.Length > 4.95 6 7.638 virginica ( 0.00000 0.33333 0.66667 ) *
  7) Petal.Width > 1.75 46 9.635 virginica ( 0.00000 0.02174 0.97826 )
     14) Petal.Length < 4.95 6 5.407 virginica ( 0.00000 0.16667 0.83333 ) *
     15) Petal.Length > 4.95 40 0.000 virginica ( 0.00000 0.00000 1.00000 ) *

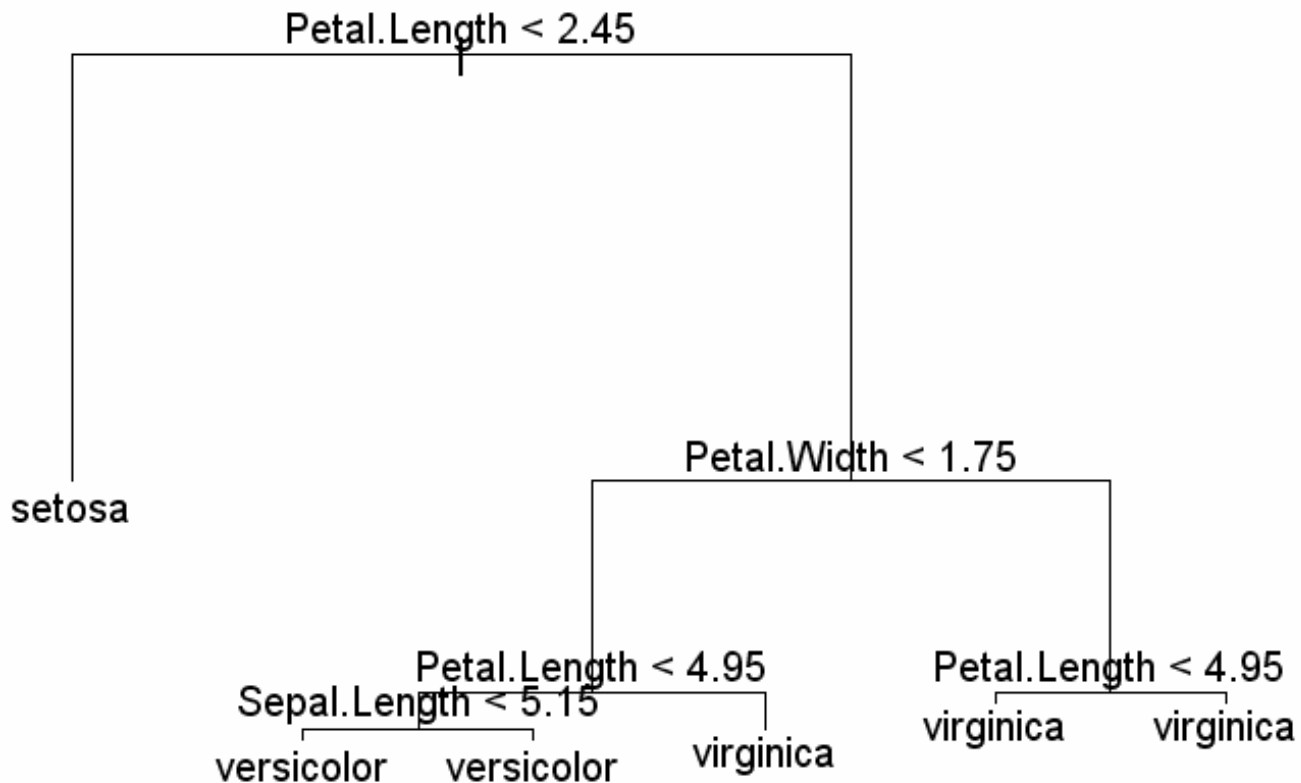
> summary(ir.tr)
```



```
Classification tree:
tree(formula = Species ~ ., data = iris)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width" "Sepal.Length"
```

# Plot the fitted tree model

- `> plot(ir.tr)`
- `> text(ir.tr)`





# Predictions from a Fitted Tree Object

```
> predict(ir.tr, iris[,1:4], type="class")
 [1] setosa      setosa      setosa      setosa      setosa      setosa
 [7] setosa      setosa      setosa      setosa      setosa      setosa
[13] setosa      setosa      setosa      setosa      setosa      setosa
[19] setosa      setosa      setosa      setosa      setosa      setosa
[25] setosa      setosa      setosa      setosa      setosa      setosa
[31] setosa      setosa      setosa      setosa      setosa      setosa
[37] setosa      setosa      setosa      setosa      setosa      setosa
[43] setosa      setosa      setosa      setosa      setosa      setosa
[49] setosa      setosa      versicolor  versicolor  versicolor  versicolor
[55] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
[61] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
[67] versicolor  versicolor  versicolor  versicolor  virginica   versicolor
[73] versicolor  versicolor  versicolor  versicolor  versicolor  virginica
[79] versicolor  versicolor  versicolor  versicolor  versicolor  virginica
[85] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
[91] versicolor  versicolor  versicolor  versicolor  versicolor  versicolor
[97] versicolor  versicolor  versicolor  versicolor  virginica   virginica
[103] virginica   virginica   virginica   virginica   versicolor  virginica
[109] virginica   virginica   virginica   virginica   virginica   virginica
[115] virginica   virginica   virginica   virginica   virginica   virginica
[121] virginica   virginica   virginica   virginica   virginica   virginica
[127] virginica   virginica   virginica   virginica   virginica   virginica
[133] virginica   virginica   virginica   virginica   virginica   virginica
[139] virginica   virginica   virginica   virginica   virginica   virginica
[145] virginica   virginica   virginica   virginica   virginica   virginica
Levels: setosa versicolor virginica
```

```
> █
```



# Computing the accuracy

---

- `> predict(ir.tr, iris[,1:4], type="class") -> ir.pred`
- `> sum(ir.pred == iris[,5])/nrow(iris)`  
`[1] 0.9733333` ← accuracy



## Package : “e1071”

---

- Install new packages for
  - `> install.packages("e1071")`
- Include “svm”, “naiveBayes”, ....  
functions
- Can tune hyperparameters of statistical methods : “tune” function



# Using svm in “e1071” (1)

```
> library(e1071); help("svm")
```

## Usage

```
## S3 method for class 'formula':  
svm(formula, data = NULL, ..., subset, na.action =  
na.omit, scale = TRUE)  
## Default S3 method:  
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =  
"radial", degree = 3, gamma = 1 / ncol(as.matrix(x)), coef0 =  
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsi  
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TR  
..., subset, na.action = na.omit)
```

## Arguments

formula	a symbolic description of the model to be fit.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which ‘svm’ is called from.
x	a data matrix, a vector, or a sparse matrix (object of class

# Using svm in “e1071” (2)

- Fitted a svm model for iris dataset

```
> model <- svm(factor(Species) ~ ., data = iris)
> summary(model)
```

Call:

```
svm(formula = factor(Species) ~ ., data = iris)
```

Parameters:

```
SVM-Type: C-classification
```

```
SVM-Kernel: radial
```

```
cost: 1
```

```
gamma: 0.25
```

Default setting

```
Number of Support Vectors: 51
```

```
( 8 22 21 )
```

```
Number of Classes: 3
```

```
Levels:
```

```
setosa versicolor virginica
```





## Using `svm` in “e1071” (3)

---

- Predict from `svm` model

```
> x<-iris[,-5]
```

```
> y<-iris[,5]
```

```
> predict(model,x)->pred
```

```
> print(pred)
```

- Compute accuracy for prediction

```
> sum(pred==y)/length(y)
```

```
[1] 0.9733333
```

# Using svm in “e1071” (4)

- Using *Linear* kernel to fit svm model


```
> model <- svm(factor(Species) ~ ., data = iris, kernel="linear")  
> summary(model)
```

Call:

```
svm(formula = factor(Species) ~ ., data = iris, kernel = "linear")
```

Parameters:

```
SVM-Type: C-classification  
SVM-Kernel: linear  
cost: 1  
gamma: 0.25
```



Number of Support Vectors: 29

```
( 2 15 12 )
```

Number of Classes: 3

Levels:

```
setosa versicolor virginica
```



## Using svm in “e1071” (5)

---

- Predict from svm *Linear* model

```
> predict(model,x)->linear.pred
```

```
> sum(linear.pred==y)/length(y)
```

```
[1] 0.9666667
```

- Confusion matrix

```
> table(linear.pred, y)
```

linear.pred	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	46	1
virginica	0	4	49

# Parameter Tuning

## Function: “tune” in “e1071” (1/2)

### ■ “tune” function in e1071

#### Usage:

```
tune(method, train.x, train.y = NULL, data = list(), validation.x =  
      NULL, validation.y = NULL, ranges = NULL, predict.func = predict,  
      tunecontrol = tune.control(), ...)  
best.tune(...)
```

#### Arguments:

method: function to be tuned.

train.x: either a formula or a matrix of predictors.

train.y: the response variable if 'train.x' is a predictor matrix.  
Ignored if 'train.x' is a formula.

data: data, if a formula interface is used. Ignored, if predictor  
matrix and response are supplied directly.

validation.x: an optional validation set. Depending on whether a  
formula interface is used or not, the response can be  
included in 'validation.x' or separately specified using  
'validation.y'.

validation.y: if no formula interface is used, the response of the  
(optional) validation set.

ranges: a named list of parameter vectors spanning the sampling  
space. The vectors will usually be created by 'seq'.

# Parameter Tuning

## Function: “**tune**” in “**e1071**” (2/2)

- tune “svm” for classification with RBF-kernel (default in svm)

- gamma=0.5,1,2 and cost=4,8,16

```
> obj <- tune(svm, Species~., data = iris,  
+ ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)))  
> obj
```

```
Parameter tuning of `svm':
```

```
- sampling method: 10-fold cross validation
```

```
- best parameters:
```

```
gamma cost  
0.5      4
```

```
- best performance: 0.04
```

# k-Nearest Neighbour

## Classification : knn in “class” (1/2)

### Usage:

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

### Arguments:

train: matrix or data frame of training set cases.

test: matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.

cl: factor of true classifications of training set

k: number of neighbours considered.

l: minimum vote for definite decision, otherwise 'doubt'. (More precisely, less than 'k-1' dissenting votes are allowed, even if 'k' is increased by ties.)

prob: If this is true, the proportion of the votes for the winning class are returned as attribute 'prob'.

use.all: controls handling of ties. If true, all distances equal to the 'k'th largest are included. If false, a random selection of distances equal to the 'k'th is chosen to use exactly 'k' neighbours.



# k-Nearest Neighbour

## Classification : **knn** in “class” (2/2)

- For example: iris dataset

```
> trn.x<-iris[,-5] ; tst.x<-iris[,-5]
```

```
> trn.y<-iris[,5] ; tst.y<-iris[,5]
```

- Fit the knn with k=3

```
> knn(trn.x, tst.x, trn.y, k = 3, prob=TRUE)->knn.pred
```

```
> print(knn.pred)
```

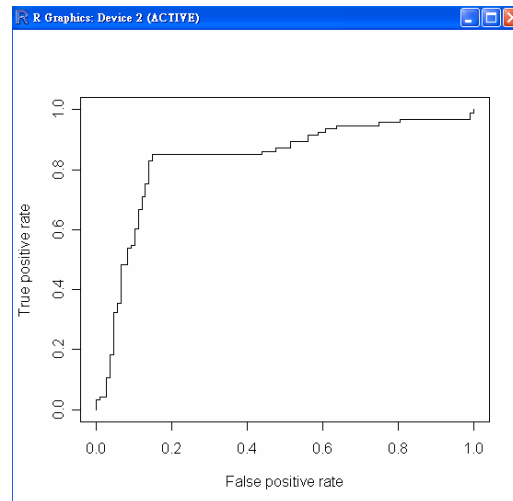
- Confusion matrix

```
> table(knn.pred, tst.y)
```

knn.pred	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	3
virginica	0	3	47

# Evaluation : Receiver Operating Characteristic (ROC) curve analysis

- Plot ROC Curve : “**ROCR**” package
  - > library(ROCR); data(ROCR.simple)
  - > pred <- prediction( ROCR.simple\$predictions, ROCR.simple\$labels)
  - > perf <- performance(pred, "tpr", "fpr")
  - > plot(perf)







# Packages for Machine learning

---

- For classification
  - tree in “tree”
  - svm in “e1071”
  - knn in “class”
  - lda in “MASS”
  - adaboost in “boost”
- For clustering
  - kmean in “stats”
- Other useful packages
  - caTools 、 kernlab 、 mlbench 、 cluster 、 ...