R is a programming language designed to support data analysis and model building.

- All traditional programming constructs such as expressions, assignments, conditionals, loops, and functions are present.
- A straight-forward object system that supports high-level constructs such as statistical models with all their parameters etc. very nicely.
- Vector arithmetic (very powerful and the preferred way of accomplishing things in R).
- Graphics engine supporting graphical techniques (automatic scatter plots, histograms, etc.)
- Many, many extension modules implementing everything from basic statistics to micro array analysis...and in particular support vector machines.
Interactive R Session

Version 2.2.1 (2005-12-20 r36812)
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2 + 2
[1] 4
> |

– p. 2/1
R Programming

```r
> x <- 2
> 2 * x
[1] 4

> v <- c(1,2,3)
> v
[1] 1 2 3

> v + 1
[1] 2 3 4

> w <- v + 1
> q <- w + v
> q
[1] 3 5 7

> add1 <- function(x) { x + 1; }
> add1
function(x) { x + 1; }
> add1(1)
[1] 2
```
This function performs the same operation as the vector operation `w + 1`. From a performance point of view it is always desirable to use the vector operations, explicit iteration over vector elements is SLOW!
R Data

R has many different ways to represent data:

- vectors
- lists
- arrays/matrices

The most important one (for our purposes) is the data frame. A data frame is a two-dimensional data matrix with additional structure.

```r
> df <- data.frame(v,w)
> df
   v w
 1 1 2
 2 2 3
 3 3 4
> df$v
[1] 1 2 3
> df$w
[1] 2 3 4
```
Loading Data Frames

We can read comma-separated-value (CSV) files directly into an R data frame.

Here is our mammal training data set represented as a CSV file:

Legs, Wings, Fur, Feathers, Mammal
4, no, yes, no, true
2, yes, no, yes, false
4, no, no, no, false
4, yes, yes, no, true
3, no, no, no, false

Assume that we saved this into a file called “mammals.csv”, in a directory called “datasets”.
Loading Data Frames

```r
> setwd("datasets")
> mammals.df <- read.csv("mammals.csv")
> mammals.df

   Legs Wings Fur Feathers Mammal
1   4    no  yes     no  true
2   2    yes no  yes  false
3   4    no  no     no  false
4   4    yes yes  no  true
5   3    no  no     no  false

> summary(mammals.df)

     Legs  Wings  Fur  Feathers  Mammal
Min.  :2.00  no :3  no :3  no :4  false:3
1st Qu.:3.00 yes:2 yes:2 yes:1  true :2
Median :4.00
Mean  :3.40
3rd Qu.:4.00
Max.  :4.00
```
For convenience sake, R comes with a number of predefined data frames. One such predefined data frame is the *iris data set*.

```r
> data(iris)
> summary(iris)

    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width         Species
      Min. :4.300  Min. :2.000  Min. :1.000  Min. :0.100          setosa :50
    1st Qu.:5.100  1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300  versicolor:50
    Median :5.800  Median :3.000  Median :4.350  Median :1.300       virginica:50
     Mean :5.843  Mean :3.057  Mean :3.758  Mean :1.199
    3rd Qu.:6.400  3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800
```

We might wish to inspect the data distributions visually as well:

```r
> hist(iris$Sepal.Length)
> hist(iris$Petal.Length)
```
R Built-in Data Frames

Histogram of Iris$Sepal.Length

Histogram of Iris$Petal.Length
R Built-in Data Frames

```r
> plot(iris)
```

![Scatter plot matrix showing relationships between variables in the iris dataset.](image)
> attach(iris)
> model <- lm(Petal.Length ~ Petal.Width)
> plot(Petal.Width, Petal.Length)
> abline(model)
Homework

Read Chapter 1 and Appendix B
Do Assignment 1, see website.