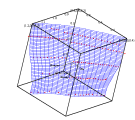
Agenda for today

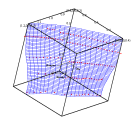
1. Descriptive Data Analysis
2. Graphics



Prerequisites

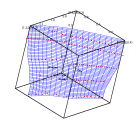
```
1  library("xplore")
2  library("stats")
3  setenv("outputstringformat","%s")
```

Avoid quotes in text output.



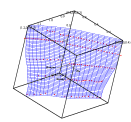
Descriptive Data Analysis

- typically the first part of statistical modeling
- evaluation of data
- All routines from libraries xplore (basic routines) and stats (basic statistical methods)



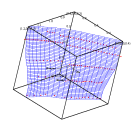
Data Matrices

- $z = \#(x_1, x_2, \ldots, x_n)$
  creates a column vector z from scalar numbers $x_1, x_2, \ldots, x_n$
- $z = x|y$
  concatenates two arrays x and y rowwise
- $z = x \sim y$
  concatenates two arrays x and y columnwise



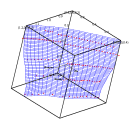
Reading Data

```
x = read("file")
    reads numeric data from file.dat
x = readm("file")
    reads mixed text and numeric data from file.dat
```

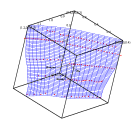
Dimensions of a Dataset

```
d = dim(x)
    shows the dimension of an array x
n = rows(x)
    shows the number of rows of an array x
p = cols(x)
    shows the number of columns of an array x
y = x[i,j] or y = x[i,] or y = x[,j]
    extracts element i,j or row i or column j from x
z = x[k:l,m:n]
    extracts rows k to l and columns m to n
```

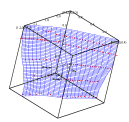
Minimum and Maximum

```
1 mx = min(x {,d})
2      computes the minimum of an array x, optionally
         with respect to dimension d
3 mx = max(x {,d})
4      computes the maximum of an array x, optionally
         with respect to dimension d
```

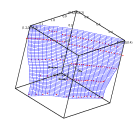
Mean, Variance and other Moments

```
1 mx = mean(x {,d})
2     computes the mean of an array x, optionally with
        respect to dimension d
3 vx = var(x {,d})
4     computes the variance of an array x, optionally
       with respect to dimension d
5 kx = kurtosis(x)
6     computes the (columnwise) kurtosis of an array x
7 sx = skewness(x)
8     computes the (columnwise) skewness of an array x
```

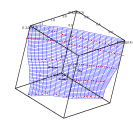
Median and Quantiles

```
mx = median(x)
     computes the (columnwise) median of an array x
qx = quantile(x, alpha)
     computes the (columnwise) quantile of an array x
        at level alpha
```

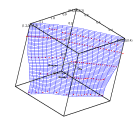
Covariance and Correlation

```
cx = cov(x)
     computes the covariance matrix of a data matrix
       x
rx = corr(x)
     computes the correlation matrix of a data matrix
       x
```

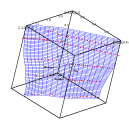
Categorical Data

```
{xr, r} = discrete(x {,y})
    reduces a matrix to its distinct rows
    and gives the number of replications of each row
    in the original data set
```

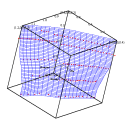
Categorical Data

```
setenv("outputstringformat","%s")
library("xplore")
library("stats")
earn=read("cps85")
earn=earn[,1|2|5|8|10|11|12]
{cat,freq}=discrete(earn[,3])
cat
freq
```

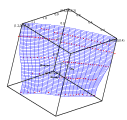
Missing Data

```
1 nx = countNaN(x)
2      counts missing values in an array x
3 nx = countNotNumber(x)
4      counts missing and infinite values in an array x
5 ix = isNaN (x)
6      determines whether the elements of an array x
         are missing values
7 ix = isInf (x)
8      determines whether the elements of an array x
         are infinite values
```

Missing Data

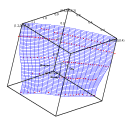
```
ix = isNumber (x)
     determines whether the elements of an array x
       are regular numeric values
y = paf(x, i)
     deletes all rows of x for which the
       corresponding element of i equals 0
y = replace(x, w, b)
     replaces all elements of x which equal w by the
       value b
```

Summarizing Information

```
s = summarize(x {,xvars})
    computes a short summary of descriptive
      statistics
s = fivenum(x {,xvars})
    computes the five number summary for each column
       of a matrix x
s = descriptive(x {,xvars})
    computes detailed descriptive statistics for
      each column of a matrix x
```

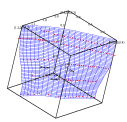
optionally a vector of variable names xvars can be given



Summarizing Metric Data

```
1 s = summarize(x {,xvars})
2     computes a short summary of descriptive
        statistics
3 s = fivenum(x {,xvars})
4     computes the five number summary for each column
        of a matrix x
5 s = descriptive(x {,xvars})
6     computes detailed descriptive statistics for
        each column of a matrix x
```

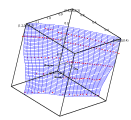
optionally a vector of variable names xvars can be given



Summarizing Categorical Data

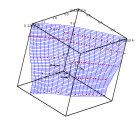
```
1 s = frequency(x {, xvars {, outwidth}})
2     computes frequency table for each column of a
        matrix x
3 s = crosstable(x{,xvars})
4     computes pairwise cross tables from all columns
5     computes the result of a $ \chi^2$ independence
        test
```

optionally a vector of variable names `xvars` can be given



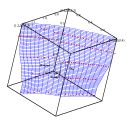
Graphics Overview

- the graphical tools (high-level): plot*
- the graphical primitives (low-level) gr*
- the graphical commands



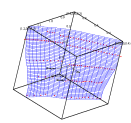
Basic Plotting

```
plot(x1 {, x2 {,...{x5}}})
    plots the data sets x1,...,x5
line(x1 {, x2 {, ... {x5}}})
    plots the lines sets x1, ..., x5
y = setmask (x, opt1 {, opt2 {, ... {opt9}}})
    modifies a data set for plotting
disp = createdisplay (r,c)
    creates a display disp
show(disp, i, j, x1 {, x2 {, ... {, xn}}})
    plots the data sets x1, ..., xn in the display
      disp d
```

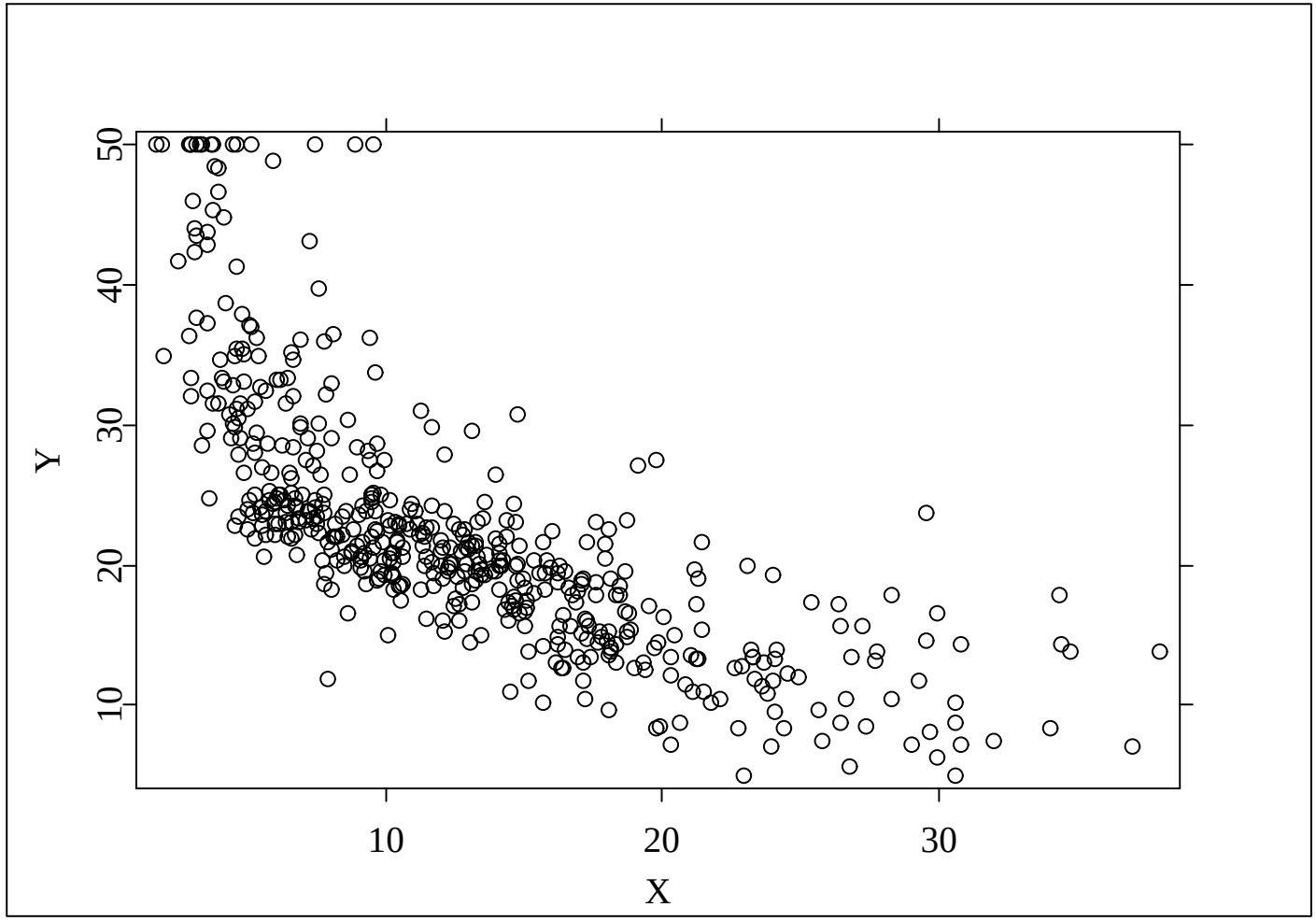



Basic Plotting

```
1 library ("plot")                ; loads library plot
2 data = read ("bostonh")         ; reads Boston Housing data
3 x = data[,13:14]                ; selects columns 13 and 14
4 plot(x)                         ; plots data set
```

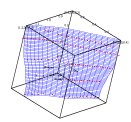



Example plot

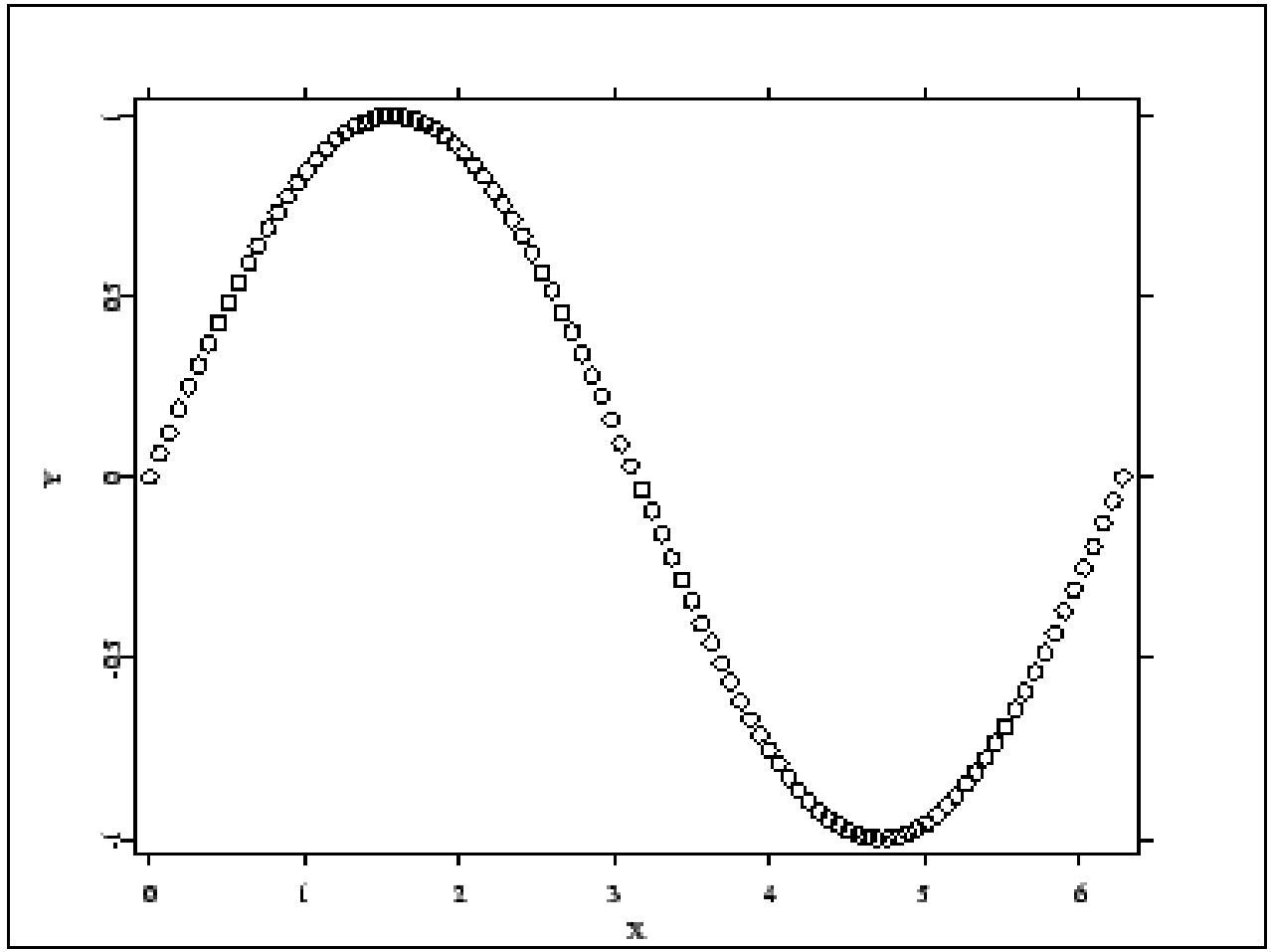


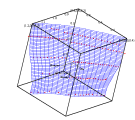
Basic Plotting

```
1 library("plot")                          ; loads library plot
2 xmin = 0                                 ; grid minimum
3 xmax = 2*pi                              ; grid maximum
4 n    = 100                               ; number of grid
    points
5 x = xmin + (xmax-xmin)/(n-1) .* (0:n-1)
6 ; generates grid
7 y = sin(x)                               ; computes sin(x)
8 plot(x~y)                                ; plots data set
```

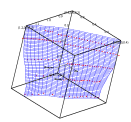
Example plot



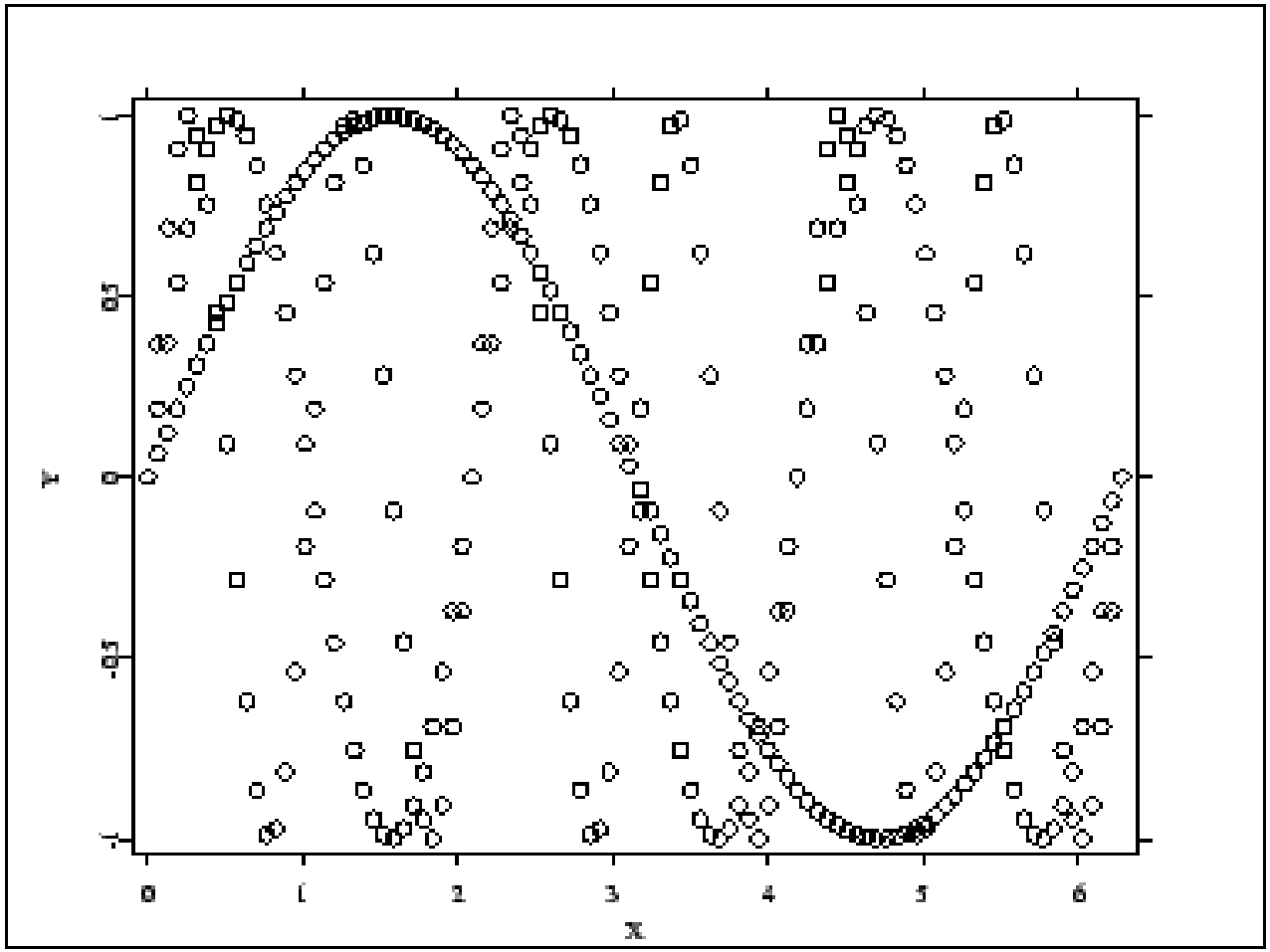


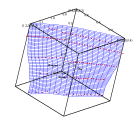
Multiple Plots

```
1  library("plot")
2  xmin = 0
3  xmax = 2*pi
4  n    = 100
5  x  = xmin + (xmax-xmin)/(n-1) .* (0:n-1)
6  y1 = sin(x)
7  y2 = sin(3.*x)
8  y3 = sin(6.*x)
9  plot(x~y1, x~y2, x~y3)
```

Example plot

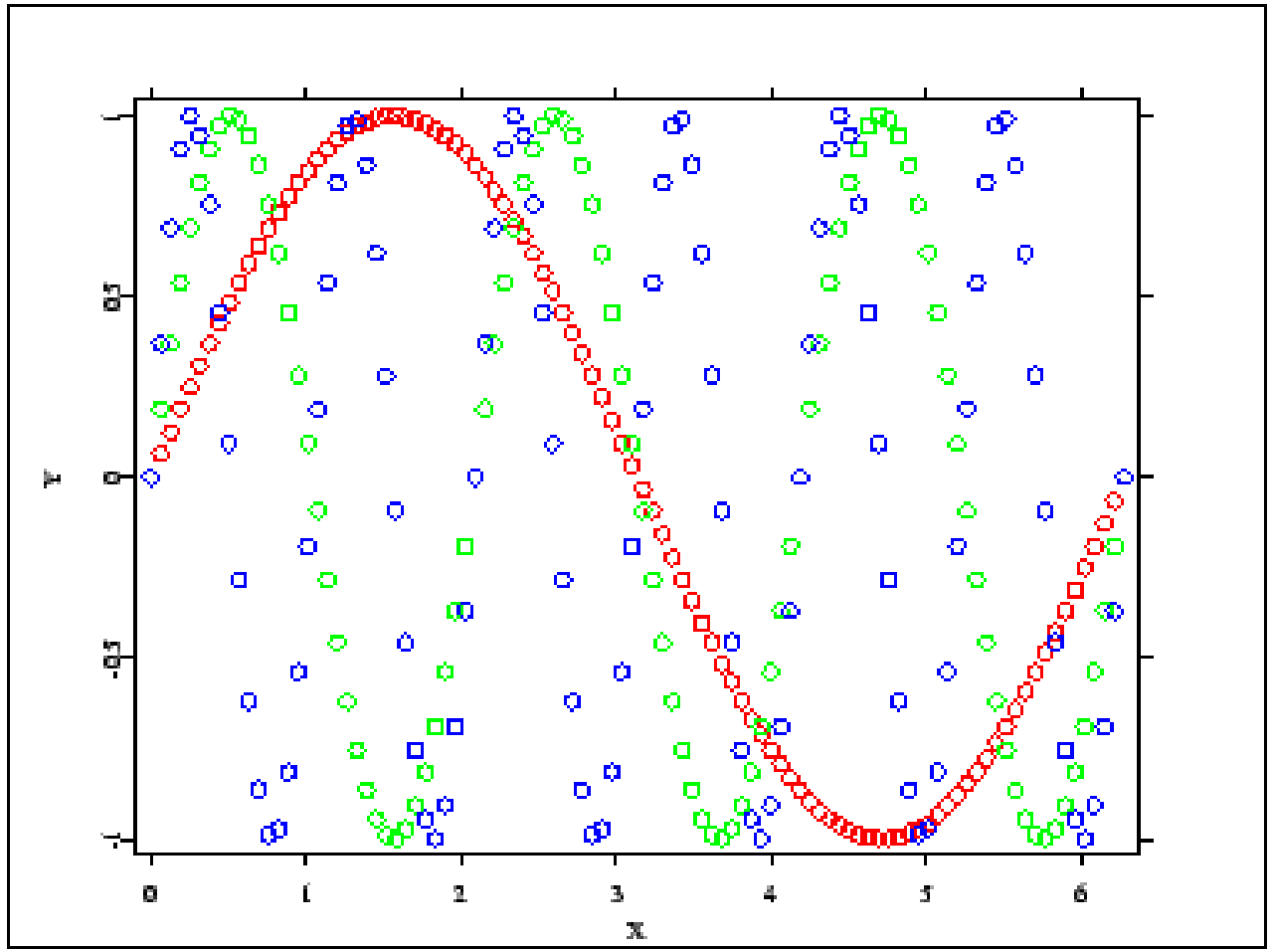


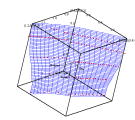


Colors

```
1   library("plot")
2   xmin = 0
3   xmax = 2*pi
4   n   = 100
5   x   = xmin + (xmax-xmin)/(n-1) .* (0:n-1)
6   y1  = sin(x)
7   y2  = sin(3.*x)
8   y3  = sin(6.*x)
9   z1  = setmask(x~y1, "red")
10  z2  = setmask(x~y2, "green")
11  z3  = setmask(x~y3, "blue")
12  plot(z1, z2, z3)
```

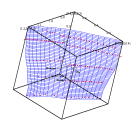
Example plot



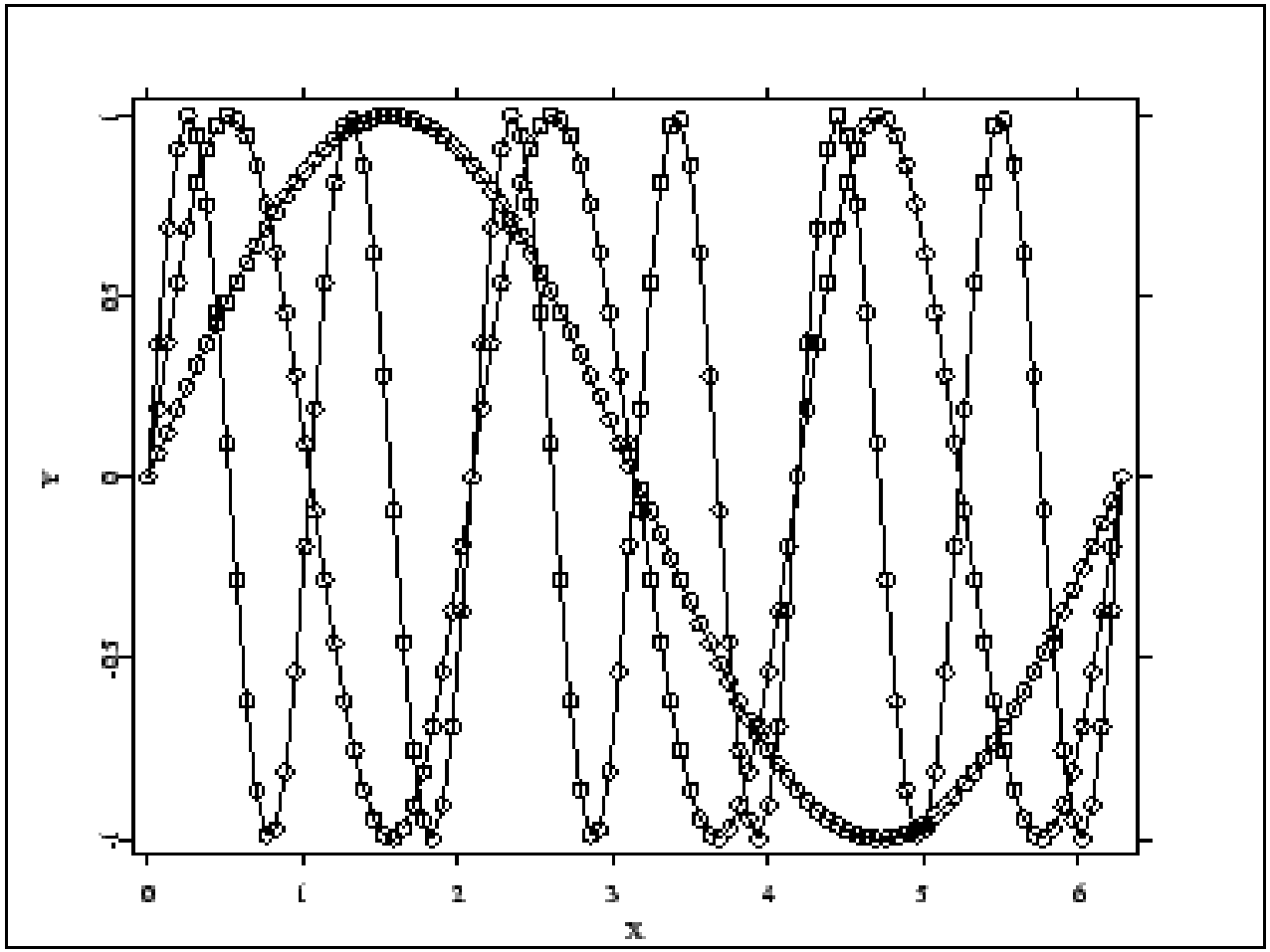


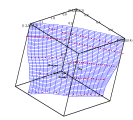
Plotting Lines

```
1  library("plot")
2  xmin = 0
3  xmax = 2*pi
4  n    = 100
5  x  = xmin + (xmax-xmin)/(n-1) .* (0:n-1)
6  y1 = sin(x)
7  y2 = sin(3.*x)
8  y3 = sin(6.*x)
9  line(x~y1, x~y2, x~y3)
```

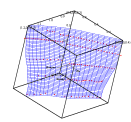
Example plot



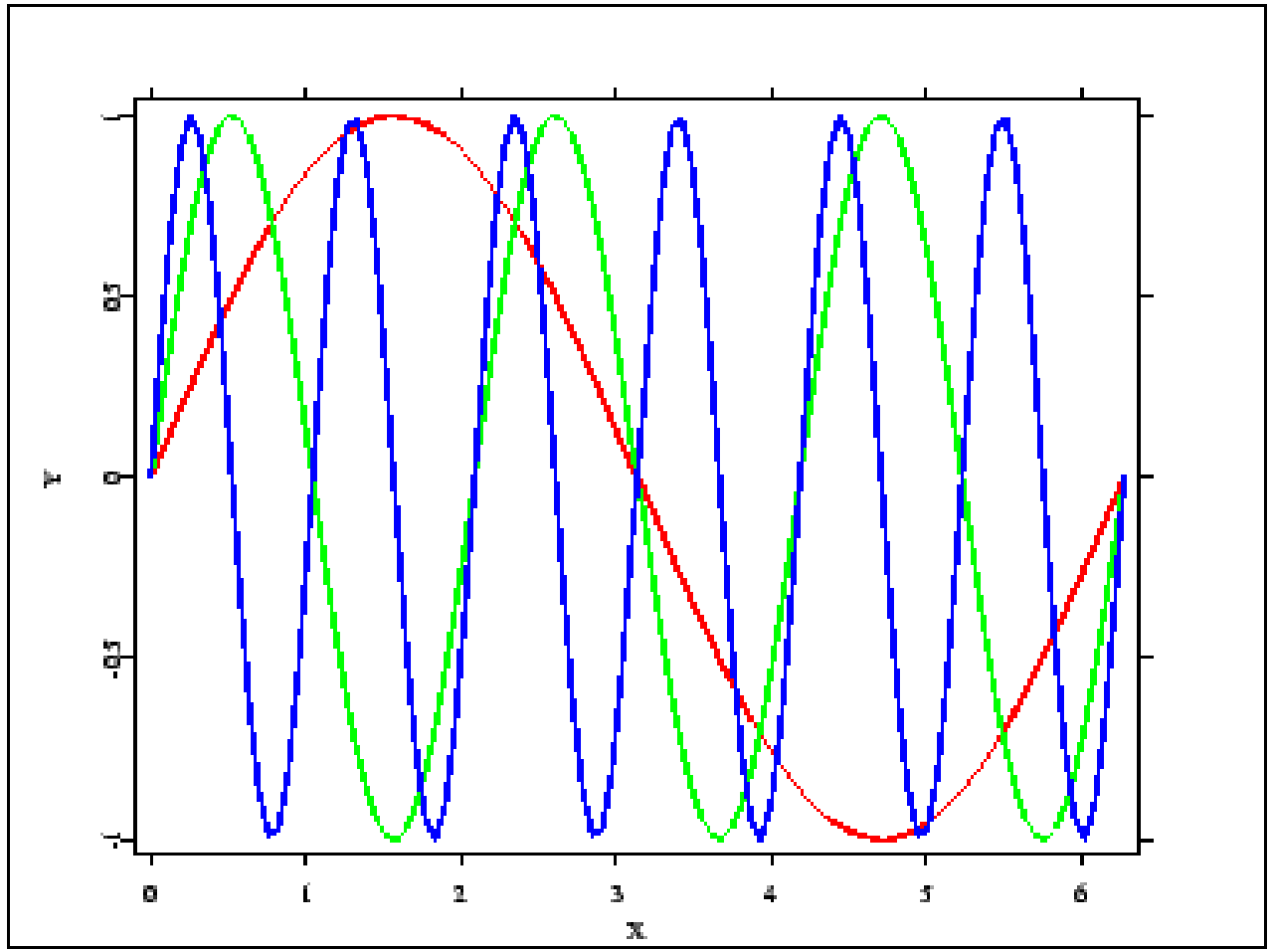


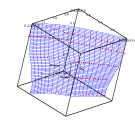
Lines and Colors

```
1   library("plot")
2   xmin = 0
3   xmax = 2*pi
4   n    = 100
5   x  = xmin + (xmax-xmin)/(n-1) .* (0:n-1)
6   y1 = sin(x)
7   y2 = sin(3.*x)
8   y3 = sin(6.*x)
9   plot(x~y1, x~y2, x~y3)
10  z1 = setmask(x~y1, "line", "red")
11  z2 = setmask(x~y2, "line", "green")
12  z3 = setmask(x~y3, "line", "blue")
13  plot(z1, z2, z3)
```

Example plot





Multiple Plots

```
1 disp = createdisplay(rownum, colnum)
2
3 show(disp,<row>,<col>, what)
```