# STAT/ME 424
# XLISPSTAT Quickstart

Professor Wei-Yin Loh
Department of Statistics
University of Wisconsin–Madison

Revised January 8, 2002

## Contents

## 1 Introduction

This document is intended to help you get started with XLISPSTAT, a public-domain computer program for statistics, data analysis and graphics. A complete guide and reference is Tierney (1990). On-line tutorials can be found at

- `http://www.stat.umn.edu/~luke/xls/tutorial/techreport/techreport.html`
- `http://landau.MINES.edu/~jscales/xls/xls.html`

XLISPSTAT runs on many computers, including:

1. Apple Macintosh/PowerMac and compatibles under Mac OS 8/9/X and LinuxPPC,
2. Intel PCs and compatibles under Microsoft Windows 3.x/9x/Me/NT/2000/XP,
3. Linux/Unix workstations under X-windows.

The program is installed on the Statistics Department's Linux workstations in Room 1321 Computer Science. If you wish to install it on your own computer, please follow the instructions in the next section.

## 2 Installing XLISPSTAT on your personal computer

### 2.1 Mac and Windows

Login to the class WebCT page and click on the icon named "`Xlispstat files`."

**Win9x/Me/NT/2000:** Download the file `xlisp.zip` to a folder on your PC and unzip it with a program such as `WinZip`.

**Mac OS 9x/X:** Download the file `xlisp.sit` to a folder on your Mac and unstuff it with a program such as `StuffIt`.

### 2.2 Linux and Unix

Obtain the C source from `http://www.xlispstat.org` and compile it.

## 3 Running the program

Double-click the program icon `Wxls32.exe` (PC) or "`XLISP-STAT 3-52-16 PPC`" (Mac) to begin. On a Linux/Unix workstation, type the name `xlispstat` to start. If your installation is successful, you should see something like the following displayed on the opening screen:

```
XLISP-PLUS version 3.04
Portions Copyright (c) 1988, by David Betz.
Modified by Thomas Almy and others.
XLISP-STAT Release 3.52.16 (Beta).
Copyright (c) 1989-1999, by Luke Tierney.

>
```

The final “>” character is the the XLISPSTAT prompt.

## 4 Exiting the program

Type

```
> (exit)
```

or use the `Quit` command in the `File` menu (Macintosh or Windows versions).

## 5 Simple calculations

In XLISPSTAT, all text following a semi-colon are regarded as comments and are ignored. These comments are given in the following examples only to explain the command. They are not necessary for execution.

```
> (+ 2 6) ; add 2 to 6
8
> (+ 1 2 3 4 5) ; add 1, 2, 3, 4, and 5
15
> (- 8 5) ; subtract 5 from 8
3
> (- 8 1 3 2) ; recursive subtraction
2
> (* 3 7) ; 3 times 7
21
> (* 3 2 4) ; multiply 3, 2 and 4
24
> (/ 6 3) ; divide 6 by 3
2
> (+ 6 (* 2 4)) ; 6+(2*4)
14
```

## 6 Mathematical functions

XLISPSTAT understands many of the standard mathematical functions.

## Elementary operations

```
> (sqrt 2)      ; square root of 2
1.41421
> (log 10)      ; logarithm to base e of 10
2.30259
> (ˆ 2 4)       ; 2 to the power 4
16
> (exp 1)       ; e raised to power 1
2.71828
> (max 2.1 5.6 7.3)     ; maximum of 2.1, 5.6 and 7.3
7.3
> (min 2.1 5.6 7.3)     ; minimum of 2.1, 5.6 and 7.3
2.1
> (abs 2)       ; absolute value of 2
2
> (abs -2)      ; absolute value of 2
2
> (rem 10 3)    ; remainder of dividing 10 by 3
1
> (round 10.4)  ; round 10.4 to nearest integer
10
> (round 10.5)
11
> (ceiling 10.5)   ; smallest integer not less than 10.5
11
> (floor 10.5)     ; largest integer not larger than 10.5
10
```

## Logical functions

```
> (> 5 3)   ; test if 5 is greater than 3
T
> (> 3 5)   ; test if 3 is greater than 5
NIL
> (= 5 4)   ; test if 5 = 4
NIL
> (>= 5 4)  ; test if 5 is greater than or equal to 4
T
> (<= 5 5)  ; test if 5 is less than or equal to 4
T
> (< 2 2.2) ; test if 2 < 2.2
T
```

### Trigonometric functions

```
> pi            ; value of Pi
3.14159
> (sin (/ pi 2))        ; sin of Pi/2 (in radians)
1
> (cos (/ pi 6))        ; cosine
0.866025
> (tan (/ pi 4))        ; tangent
1
> (asin 1)     ; arcsine of 1 in radians
1.5708
> (acos 0)     ; arcosine
1.5708
> (atan 1)     ; arctangent
0.785398
```

### Operations on lists

XLISPSTAT can add lists of numbers component-wise. The following example illustrates addition.

```
> (+ (list 1 2 3) (list 4 5 6))
(5 7 9)
```

## 7 Basic statistical functions

### 7.1 Statistical distributions and random numbers

```
> (binomial-cdf 10 15 0.4) ; binomial cdf at 10 for 15 trials and p=0.4
0.990652
> (binomial-pmf 10 15 0.4) ; binomial prob. for 10 successes
0.0244856
> (chisq-cdf 5 4)       ; value of chi-square cdf at 5 with 4 df
0.712702
> (chisq-quant 0.05 4)  ; 0.05-quantile of chi-square dist with 4 df
0.710723
> (chisq-rand 3 4)      ; generates 3 random chi-square values with 4 df
(8.0844 1.6684 3.70551)
> (f-cdf 2 3 5) ; cdf of F dist at 2 with 3 and 5 df
0.767376
> (f-quant 0.95 2 6)    ; upper 5% point of F dist with 2 and 6 df
5.14308
> (normal-cdf 1.0)      ; cdf of standard normal dist at 1.0
0.841345
```

```
> (normal-dens 0)       ; standard normal density at 0
0.398942
> (normal-quant 0.05)   ; lower 5% point of standard normal dist
-1.64485
> (normal-rand 5)       ; 5 standard normal random numbers
(-0.550957 0.429603 1.34749 2.59364 -0.243258)
> (poisson-cdf 3 2)     ; Poisson cdf at 3 with mean 2
0.857123
> (poisson-rand 3 2)    ; 3 Poisson random numbers with mean 2
(3 2 3)
> (t-cdf 5 3)           ; cdf at 5 of t dist with 3 df
0.992304
> (t-quant 0.95 5)      ; upper 5% point of t dist with 5 df
2.01505
> (uniform-rand 4)      ; 4 random uniform numbers on (0, 1)
(0.934608 0.329246 0.556881 0.0739395)
```

### 7.2 Elementary data management functions

```
> (difference (list 1 2 4 7 9)) ; first differences for a sequence
(1 2 3 2)
> (mean (list 1 2 4 7 9));      ; mean for the list of numbers
4.6
> (interquartile-range (list 1 2 4 7 9)) ; interquartile range
5
> (median (list 1 2 4 7 9))     ; median of list
4
> (standard-deviation (list 1 2 4 7 9))
3.36155
> (rank (list 1 2 4 7 9))  ; rank (starting with 0) of each number
(0 1 2 3 4)
> (sort-data (list 1 4 2 7 9))  ; sort in ascending order
(1 2 4 7 9)
```

## 8 Defining variables

Variables may be defined using the `def` command:

```
> (def x (list 1 3 4 2 9))
X
> x            ; recall x
(1 3 4 2 9)
> (def y (list 3 1 5 6 2))
Y
```

```
> y
(3 1 5 6 2)
> (length x)    ; returns the number of elements in x
5
> (length y)
5
```

## 9 Generating and modifying data

Here are some ways to generate systematic data and to manipulate them:

```
> (def x1 (iseq 1 10))  ; generate a variable x1
X1
> x1
(1 2 3 4 5 6 7 8 9 10)
> (def y1 (rseq -10 7 10)) ; 10 equally spaced numbers between -10 and 7
Y1
> y1
(-10 -8.11111 -6.22222 -4.33333 -2.44444 -0.555556 1.33333
3.22222 5.11111 7)
> (repeat (list 1 2 3) 2)       ; repeat list twice
(1 2 3 1 2 3)
> (repeat (list 1 2 3) (list 2 3 0)) ; repeat 1 twice, 2 thrice, 3 none
(1 1 2 2 2)
> (select x1 0) ; select the first element in x1
1
> (select x1 (list 1 3))        ; select 2nd and 4th elements in x1
(2 4)
> (remove 2 x1)                 ; remove element "2" in x1
(1 3 4 5 6 7 8 9 10)
> (append x1 y1)                ; combine x1 and y1
(1 2 3 4 5 6 7 8 9 10 -10 -8.11111 -6.22222 -4.33333 -2.44444
-0.555556 1.33333 3.22222 5.11111 7)
```

## 10 Importing text data files

The command `read-data-columns` may be used to read an ASCII (plain text) data file in which each row represents one case and each column a variable. At least one space or tab should separate data values. For example, consider a file called `raw.dat` that contains the following five columns of numbers:

```
2 5 2 7 2
```

```
2 1 2 3 4
1 2 3 2 3
5 4 3 2 3
2 3 1 5 4
```

To read these data into a variable called z, type:

```
> (def z (read-data-columns "raw.dat" 5))
> z
((2 2 1 5 2) (5 1 2 4 3) (2 2 3 3 1) (7 3 2 2 5) (2 4 3 3 4))
```

The individual columns can then be extracted and given names. For example, to name the first and second columns x1 and x2, type:

```
> (def x1 (select z 0))
X1
> x1
(2 2 1 5 2)
> (def x2 (select z 1))
X2
> x2
(5 1 2 4 3)
```

## 11 Statistical graphics

The variables x and y in the following commands are two lists of numbers of the same length.

```
> (plot-points x y :title "My plot"
   :variable-labels (list "x name" "y name")) ; x-y plot
> (plot-lines x y :title "My line plot"
   :variable-labels (list "x" "y")) ; connected line plot
> (histogram x :title "My histogram") ; histogram of x values
> (boxplot x :title "My box plot")    ; boxplot of x values
> (boxplot (list x y))  ; parallel boxplots
> (probability-plot x :title "Normal PP plot")
> (quantile-plot x :title "Normal QQ plot")
> (def z (list 1 6 2 3 4))           ; add another variable
> (scatterplot-matrix (list x y z)) ; scatterplot matrix of x, y, z
> (spin-plot (list x y z))      ; 3-D plot with spinning
```

Each of these plot functions produces a window in which the plot is displayed. In the UNIX version, two buttons appear at the top of each plot window. The left button is used to close the plot window. The right button provides many options including

1. dynamically brushing and selecting scatterplots;
2. changing the size of the brush;

3. linking plots together for brushing and selection;
4. hiding selected points and re-scaling plots;
5. changing the number of bins in a histogram;
6. saving a plot as a postscript file which can be printed as usual using `lpr`.

The Mac and Windows versions are similar except that the menu bar appears at the top of the screen.

### Warnings

- In the UNIX version, the size of a printed plot is the same as that on the screen. So be careful when you resize a plot on the screen (by using the mouse).
- In the Mac and Windows versions, it is better to resize each plot within the program before pasting into a word-processor. Resizing inside a word-processor often results in jaggies.
- The default for `spin-plot` is white on black. If you wish to print one of these, you should change it (using the right menu button) to black-on-white. Otherwise you'll get a page of black ink.

## 12 Function plots

Plots like those in Box et al. (1978, pp. 296 and 297) are quite easy to produce in XLISPSTAT. For example, to plot the function

$$f(x) = 5 + 2x - x^2 \tag{1}$$

over the range $-1 < x < 2$, we can use the XLISPSTAT commands

```
> (defun f (x) (+ 5 (* 2 x) (* -1 (ˆ x 2))))
> (plot-function #'f -1 2)
```

The first line defines the function $f$ and the second makes the plot shown in Figure 1.

Contour and 3-D plots can be drawn with the `contour-function` and `spin-function` commands. For example, to graph the function

$$f(x, y) = 2 - 2x^2 - y^2 \tag{2}$$

we can use the commands

```
> (defun f (x y) (- 2 (* 2 (ˆ x 2)) (ˆ y 2)))
> (contour-function #'f -3 1 -5 -2)
> (spin-function #'f -3 1 -5 -2)
```

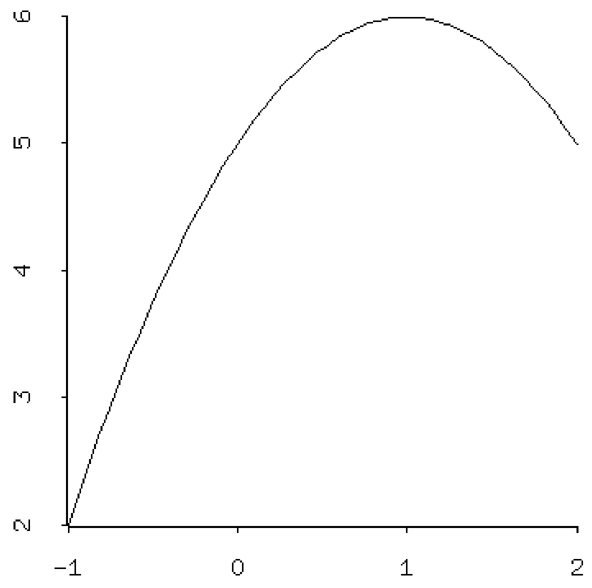

Figure 1: Plot of (1) for $-1 < x < 2$

Again, the first command line defines the function `f`. The second line draws the contour plot in Figure 2 over the region $-3 < x < 1$ and $-5 < y < -2$. The third line produces the 3-D plot in Figure 3. The latter is a *dynamic graphic*: it can be rotated in any direction by using the mouse to select one of the control buttons at the bottom of Figure 3. [Note: Figure 3 is white-on-black by default. To print the figure as shown, you have to change it to black-on-white. This is done via the `Spinner -> Options` menu.]

## 13 Low memory situations on personal computers

You will get a warning if the memory on your computer runs low. This usually occurs if you have defined too many variables or have too many plots lying on the screen. The first thing to do is to close or save unneeded plots. If this is not enough, you have to get rid of some of the defined variables. This is done using the `undef` command.

```
> (variables)   ; to see what variables are defined
(X Y)
> (undef 'x)    ; remove the variable x. QUOTE needed
X
> x
error: unbound variable - X
```

The above error message tells you that the system no longer knows about the existence of `x`.

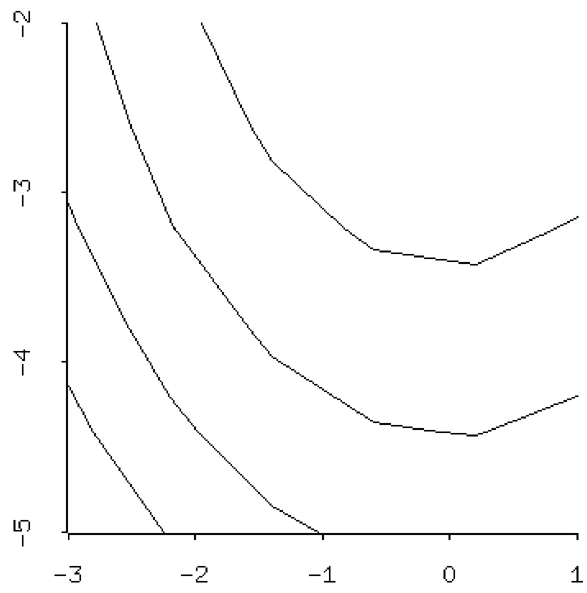

Figure 2: Contour plot of (2) for $-3 < x < 1$ and $-5 < y < -2$

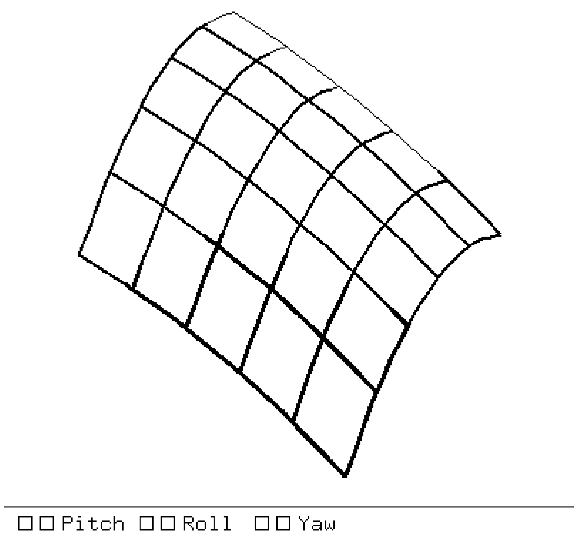

Figure 3: Spin plot of (2) for $-3 < x < 1$ and $-5 < y < -2$

## 14 Getting help

Online help is available through the `help` and `help*` commands.

```
> (help ’median)      ; if you know the exact name of the function
MEDIAN                                                [function-doc]
Args: (x)
Returns the median of the elements of X.
NIL
> (help* ’norm) ; if you are not sure about the name
------------------------------------------------------------------
BIVNORM-CDF                                           [function-doc]
Args: (x y r)
Returns the value of the standard bivariate normal distribution function
with correlation R at (X, Y). Vectorized.
------------------------------------------------------------------
Sorry, no help available on NORM
------------------------------------------------------------------
Sorry, no help available on NORMAL
------------------------------------------------------------------
NORMAL-CDF                                            [function-doc]
Args: (x)
Returns the value of the standard normal distribution function at X.
Vectorized.
------------------------------------------------------------------
NORMAL-DENS                                           [function-doc]
Args: (x)
Returns the density at X of the standard normal distribution. Vectorized.
------------------------------------------------------------------
NORMAL-QUANT                                          [function-doc]
Args (p)
Returns the P-th quantile of the standard normal distribution. Vectorized.
------------------------------------------------------------------
NORMAL-RAND                                           [function-doc]
Args: (n)
Returns a list of N standard normal random numbers. Vectorized.
------------------------------------------------------------------
NIL
```

## 15 Saving and printing

The XLISPSTAT program does not have a print function. To print text and graphics, first copy them into a word processor and then print from there. Graphs should be resized in XLISPSTAT, not in the word processor. On UNIX computers, graphs can be saved into files in PostScript format.

## 15.1 Automatic recording of text output

Text output in the listener window can be automatically recorded into a file by invoking the `dribble` command. This can be invoked either by selecting the `Dribble` item in the Mac `Command` menu or the Windows `File` menu, or by typing:

```
> (dribble "myfile")    ; save in the file called "myfile"
```

All text output will now be recorded in the file named `myfile`. To stop recording, select `dribble` again or type

```
> (dribble)             ; stops the recording
```

## 15.2 Saving data and variables

There are two ways to save data from XLISPSTAT:

**Save in LISP format:** To save a copy of defined variables in a form that can be loaded into XLISP-STAT later, do:

```
> (savevar ’(x y) "mydata") ; save data in file "mydata" (X Y)
```

This is a plain text file that can be edited by any word-processor. The data and variable names in the file can be loaded back into the program later using the `Load` command in the `File` menu.

**Save in columnar format:** Sometimes you may need to write the data in a form that can be imported by other statistics, database or spreadsheet programs. This requires that each row correspond to one case and each column to a variable. For example, if `x1`, `x2` and `x3` are three lists of the same length and you wish to write the data into a file called `filename`, first load the file `writedat.lsp` from the `STAT424` folder. Then type:

```
> (write-data-columns (list x1 x2 x3) "filename")
```

## 15.3 Controlling the number of decimals displayed

XLISPSTAT prints and saves data with full precision (up to 18 digits in some cases) by default. The default precision can be changed with the `*float-format*` command as in the following examples.

```
> *float-format*
NIL
> pi
3.141592653589793
> (setf *float-format* "%g")
"%g"
> pi
```

```
3.14159
> (setf *float-format* "%6.4g")
"%6.4g"
> pi
 3.142
> (setf *float-format* "%6.1f")
> pi
   3.1
```

## 16 Batch files

Commands that are stored in a file can be executed by XLISPSTAT with the `load` command (or with the `FILE -> LOAD` menu item in Windows and Macintosh). If the name of the command file is `tmp.lsp` (its name must end with `.lsp`), the command `(load "tmp")` will execute all the commands in it. Note, however, that numerical results are not print to the screen by default, although graphics commands will draw each plot in its own window. The command `print` can be used to print text strings and numerical results to the screen.

For example, suppose `tmp.lsp` consists of the lines

```
(def x (list 1 2 3 4 5))
(print "x =")
(print x)
(print "Standard deviation of x =")
(print (standard-deviation x))
```

Then upon loading the file, the screen will show:

```
> (load "tmp")
; loading tmp.lsp

"x ="
(1 2 3 4 5)
"Standard deviation of x ="
1.5811388300841898
```

## 17 One-way analysis of variance

The XLISPSTAT `oneway-model` function is used to perform a one-way ANOVA analysis. Below is an example session log for the blood coagulation data in Box et al. (1978, Table 6.1, p. 166). Some points to observe as you compare the results here with those in the text:

- The results from `oneway-model` are saved in an *object* called `anova` (the name is arbitrary). This allows the residuals and fitted values to be retrieved by sending *messages* to the `anova` object. We use this facility to draw the two residual plots.

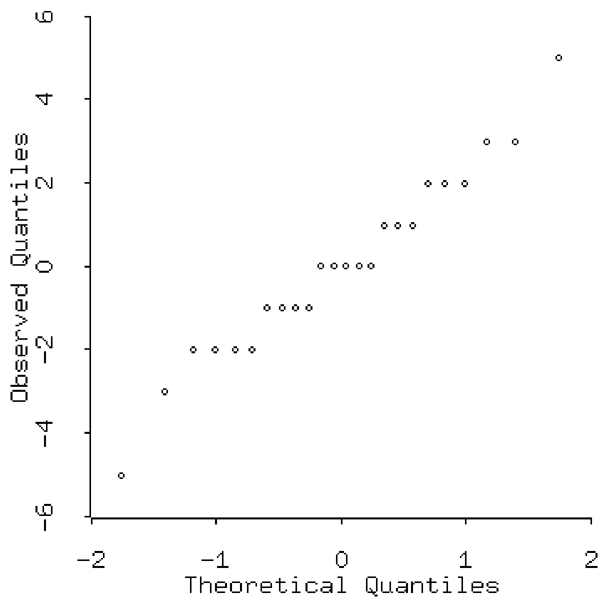

Figure 4: Normal quantile plot of residuals

- The normal quantile plot (produced by the function `quantile-plot`) and the plot of residuals versus fitted values (produced by the function `plot-points`) are shown in Figure 4 and Figure 5, respectively.
- The sample treatment means are given under the heading `Least Squares Estimates`. The number in parenthesis beside each value is the estimated standard deviation of the mean. These can be used to compute rough confidence intervals. For example, a rough 95% confidence interval for the mean of treatment 1 is $61 \pm 2 \times 1.18322$.
- The value of the $F$ statistic is saved in the variable `ftest`.
- The `oneway-model` function calculates the mean squares and degrees of freedom only. You have to multiply them to obtain the sums of squares for the ANOVA table. The values of $s_T^2$ and $s_R^2$ are given by `Group Mean Square` and `Error Mean Square`, respectively. `Sigma hat` is literally $\hat{\sigma}$, i.e., an estimate of $\sigma$. Its value is given by $s_R$, the square root of $s_R^2$.

```
> (def a (list 62 60 63 59))
> (def b (list 63 67 71 64 65 66))
> (def c (list 68 66 71 67 68 68))
> (def d (list 56 62 60 61 63 64 63 59))
> (def anova (oneway-model (list a b c d)))

Least Squares Estimates:
```

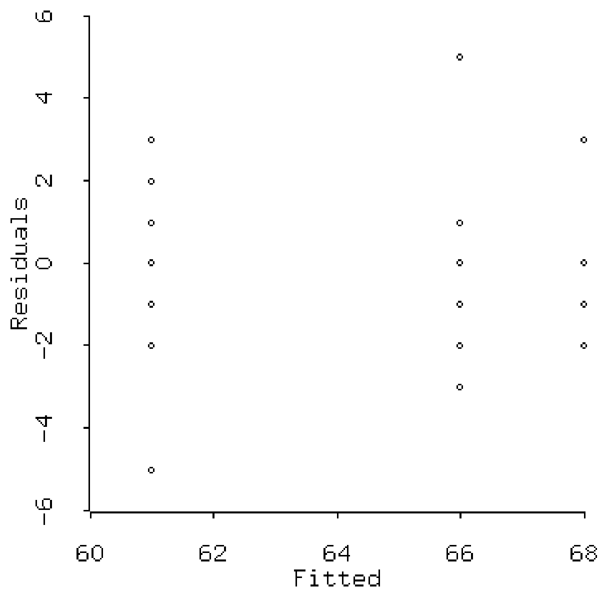

Figure 5: Plot of residuals versus fitted values

```
Group 0                         61   (1.18322)
Group 1                         66   (0.966092)
Group 2                         68   (0.966092)
Group 3                         61   (0.83666)

R Squared:                0.670588
Sigma hat:                 2.36643
Number of cases:                24
Degrees of freedom:             20

Group Mean Square:              76   (3)
Error MeanSquare:              5.6   (20)

> (def ftest (/ (send anova :group-mean-square)
                (send anova :error-mean-square)))
> ftest
13.5714
> (- 1 (f-cdf ftest (send anova :group-df)
                    (send anova :error-df)))
4.65847e-05
> (quantile-plot (send anova :residuals)
                 :title "Quantile plot of residuals")
> (plot-points (send anova :fit-values)
               (send anova :residuals)
               :title "Residuals vs fitted values"
               :variable-labels
                        (list "Fitted" "Residuals"))
```

## 18 Regression

The command for performing simple and multiple linear regression is `regression-model`.

```
> (def x (list 1 2 3 2 3 4 3 4 5 6))
> (def y (list 1 2 1 2 2 3 3 4 4 4))

> (def model (regression-model x y))  ; regress y on x

Least Squares Estimates:

Constant                  0.432836      (0.547122)
Variable 0                0.656716      (0.152331)

R Squared:                0.699085
Sigma hat:                0.682948
Number of cases:                10
Degrees of freedom:              8

MODEL

> (send model :help)  ; print keywords
loading in help file information - this will take a minute ...done
REGRESSION-MODEL-PROTO
Normal Linear Regression Model
Help is available on the following:

ADD-METHOD ADD-SLOT BASIS CASE-LABELS COEF-ESTIMATES
COEF-STANDARD-ERRORS COMPUTE COOKS-DISTANCES DELETE-DOCUMENTATION
DELETE-METHOD DELETE-SLOT DF DISPLAY DOC-TOPICS DOCUMENTATION
EXTERNALLY-STUDENTIZED-RESIDUALS FIT-VALUES GET-METHOD HAS-METHOD
HAS-SLOT HELP INCLUDED INTERCEPT INTERNAL-DOC ISNEW LEVERAGES
METHOD-SELECTORS NEW NUM-CASES NUM-COEFS NUM-INCLUDED OWN-METHODS
OWN-SLOTS PARENTS PLOT-BAYES-RESIDUALS PLOT-RESIDUALS
PRECEDENCE-LIST PREDICTOR-NAMES PRINT PROTO R-SQUARED RAW-RESIDUALS
REPARENT RESIDUAL-SUM-OF-SQUARES RESIDUALS RESPONSE-NAME RETYPE
SAVE SHOW SIGMA-HAT SLOT-NAMES SLOT-VALUE STUDENTIZED-RESIDUALS
SUM-OF-SQUARES SWEEP-MATRIX TOTAL-SUM-OF-SQUARES WEIGHTS X X-MATRIX
XTXINV Y
```

```
> (send model :residuals)  ; print residuals
(-0.08955223880597063 0.2537313432835817 -1.4029850746268657
0.2537313432835817 -0.4029850746268657 -0.059701492537313605
0.5970149253731343 0.9402985074626864 0.2835820895522385
-0.3731343283582085)

> (send model :fit-values)        ; print fitted values
(1.0895522388059706 1.7462686567164183 2.4029850746268657
1.7462686567164183 2.4029850746268657 3.0597014925373136
2.4029850746268657 3.0597014925373136 3.7164179104477615
4.3731343283582085)

> (send model :plot-residuals)    ; plot residuals versus fitted values
> (send model :plot-residuals x)  ; plot residuals versus x
> (send model :plot-residuals (ˆ x 2)) ; plot residuals versus xˆ2
> (quantile-plot (send model :residuals)) ; quantile plot of residuals

> (def model2 (regression-model x y :intercept nil))  ; exclude intercept

Least Squares Estimates:

Variable 0                0.767442      (5.886711E-2)

R Squared:                0.675544
Sigma hat:                0.668602
Number of cases:                10
Degrees of freedom:              9

MODEL2

> (def model3 (regression-model (list x (ˆ x 2)) y)) ; fit parabola

Least Squares Estimates:

Constant                  0.194444      (1.19477)
Variable 0                0.819444      (0.730229)
Variable 1                -2.314815E-2  (0.101279)

R Squared:                0.701314
Sigma hat:                0.727393
Number of cases:                10
Degrees of freedom:              7
```

```
MODEL3
```

## 19 Functions for STAT/ME 424

The functions described in this section are written specifically for STAT/ME 424. Their files are located in a subfolder called `Stat424`.

### 19.1 One and two-sample $t$ tests and intervals

The file `ttests.lsp` contains four functions for performing $t$ tests and confidence intervals.

#### 19.1.1 One-sample $t$ test and interval

The functions `one-sample-t-test` and `one-sample-t-interval` perform the procedures in their names. For example, to test that a sample y has true mean 0,

```
> (def y (list 3 8 4 2 6))
> (one-sample-t-test y)
Sample mean = 4.6000
Sample standard deviation = 2.4083
Null hypothesis is that true mean = 0
One-sample t-statistic = 4.2710
Degrees of freedom = 4
One-sided significance probability = 6.470E-3
```

To test the null hypothesis that the mean is a non-zero number, e.g., 5, simply add the value as a second argument to the function:

```
> (one-sample-t-test y 5)
Sample mean = 4.6000
Sample standard deviation = 2.4083
Null hypothesis is that true mean = 5
One-sample t-statistic = -0.37139
Degrees of freedom = 4
One-sided significance probability = 0.365
```

To find a $t$ confidence interval for the mean, use the function `one-sample-t-interval`. This produces a plot of the data with a smoothed histogram superimposed (see Figure 6). In addition, a slider dialog box allows the user to choose the confidence level of the interval. Each selection draws the interval as a horizontal line on the plot and prints the interval in the main window.

```
> (one-sample-t-interval y)
71% confidence interval = (3.2877, 5.9123)
76% confidence interval = (3.1148, 6.0852)
81% confidence interval = (2.9019, 6.2981)
```

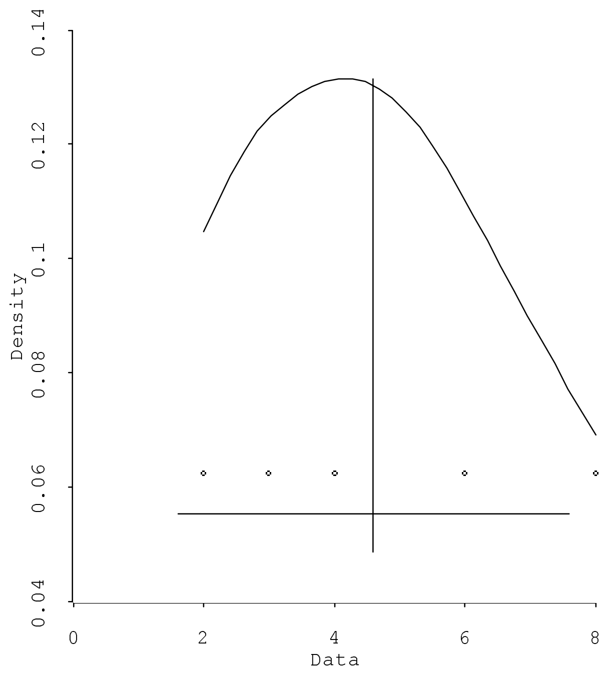

Figure 6: Smoothed histogram with plot of data values. The horizontal line is a $t$ interval for the mean. The vertical line marks the sample mean.

### 19.1.2 Two-sample $t$ test and interval

Two corresponding functions are available for the two-sample problem.

```
> (def a (list 1 2 4 5 7))
> (def b (list 5 4 6 8))
> (two-sample-t-test b a 3)
1st sample mean = 5.7500
2nd sample mean = 3.8000
Difference between sample means = 1.9500
1st sample standard deviation = 1.7078
2nd sample standard deviation = 2.3875
Pooled sample standard deviation = 2.1230
Null hypothesis is that true difference = 3
Two-sample t-statistic = -0.73728
Degrees of freedom = 7
One-sided significance probability = 0.242
```

Again, an additional argument can be included in the command to test for a nonzero mean difference.

```
> (two-sample-t-test b a 2)
1st sample mean = 5.7500
```

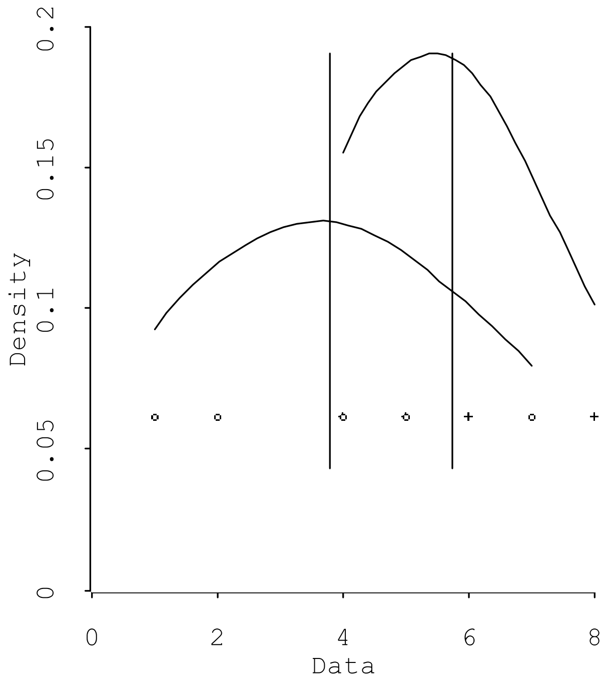

Figure 7: Smoothed histograms and data values from two samples. The vertical lines mark the two sample means.

```
2nd sample mean = 3.8000
Difference between sample means = 1.9500
1st sample standard deviation = 1.7078
2nd sample standard deviation = 2.3875
Pooled sample standard deviation = 2.1230
Null hypothesis is that true difference = 2
Two-sample t-statistic = -3.51086E-2
Degrees of freedom = 7
One-sided significance probability = 0.486
```

A $t$-interval for the difference in means obtained given by:

```
> (two-sample-t-interval b a)
75% confidence interval = (0.16371, 3.7363)
80% confidence interval = (-6.50694E-2, 3.9651)
85% confidence interval = (-0.35228, 4.2523)
90% confidence interval = (-0.74817, 4.6482)
```

In addition to the intervals, the data and their smoothed histograms are plotted in a window (see Figure 7).

## 19.2 Twoway ANOVA

The command `twoway-anova` in the file `twoway.lsp` performs analysis of variance of randomized blocks designs and two-way factorial designs.

To illustrate, the penicillin data for a randomized block design (Box et al., 1978, Table 7.1, p. 209) are analyzed next. The function gives the mean squares for the ANOVA table, the P-values for testing treatment and block effects, and the residuals under the additive model. In addition, four diagnostic plots are produced:

1. Plot of residuals versus predicted values,
2. Plot of residuals versus levels of Factor 1,
3. Plot of residuals versus levels of Factor 2,
4. Normal quantile plot of the residuals.

When selected, the points in the residual plots show their labels, i.e., their order starting from 0.

The function takes as argument a list of three lists: a list of treatment labels, a list of block labels, and a list of yields. The latter must be in the last position.

```
> (load "twoway")
> (def block (list 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5))
> (def treat (list 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4))
> (def y (list 89 88 97 94 84 77 92 79 81 87 87 85 87 92 89 84 79 81
             80 88))
> (twoway-anova (list treat block y))

ANOVA Table Assuming Additivity (Degrees of Freedom in Parentheses):

Factor 1 Mean Square:    23.33333333       (3)
Factor 2 Mean Square:           66.0       (4)
Error Mean Square:    18.83333333333       (12)
Total (Corr.) Mean Square:    29.474       (19)

P-value for Factor 1:     .3386581162
P-value for Factor 2:     4.0746173184E-2

Residuals under additive model are:
(-1 -3 2 2 3 -5 6 -4 -2 3 -1 0 1 5 -2 -4 -1 0 -5 6)
```

If there are replicated observations, two lists of residuals and their plots (corresponding to additive and non-additive models) are given.

The command `(tukey x)` prints an ANOVA table with a line for Tukey's test of transformable non-additivity. This function should be used *only* for randomized blocks designs without replications.

```
> (tukey (list treat block y))

ANOVA Table for Tukey Test of Additivity
(Degrees of Freedom in Parentheses):

Factor 1        Mean Square:    23.33333333       (3)
Factor 2        Mean Square:           66.0       (4)
Transf. Nonadd. Mean Square:    2.001082251       (1)
Remainder       Mean Square:    20.36353798       (11)
Total (Corr.)   Mean Square:    29.47368421       (19)

P-value for Tukey test of additivity: 0.7597822412573774
```

## 19.3 Latin squares

The file `latin.lsp` may be used to analyze data from a latin square, Graeco-latin square, or hyper-Graeco-latin square experiment. The following example uses the data in Box et al. (1978, Table 8.1, p. 247):

```
> (load "latin")
; loading "latin.lsp"
; loading "/usr/stat/lib/xlispstat/oneway.fsl"
T
> (def y (list 21 26 20 25 23 26 20 27 15 13 16 16 17 15 20 20))
Y
> (def drivers (list 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4))
DRIVERS
> (def cars (list 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4))
CARS
> (def additives (list 1 2 4 3 4 3 1 2 2 4 3 1 3 1 2 4))
ADDITIVES
> (latin (list drivers cars additives y))

ANOVA Table for Latin Square and Related Designs

            Mean Square     DF
Factor 1     71.999999999999986          3
Factor 2         8.          3
Factor 3     13.333333333333334          3
Error        2.6666666666666665          6
Total (corr.)19.733333333333331         15

The residuals are:
(1.0 1.0 -1.0 -1.0 1.0 1.0 -1.0 -1.0 -1.0 -1.0 1.0
1.0 -1.0 -1.0 1.0 1.0)
```

Residual plots (not shown here) are produced automatically. Additional block factors (e.g., in Graeco-latin squares) may be included in the list of arguments in the function. However, the list of observed responses (y) must be the last argument.

## 19.4 Analysis of $2^k$ factorial designs

Only one file is needed—`twolevel.lsp`. Following is a sample XLISPSTAT session in which a $2^3$ factorial experiment is analyzed. The data are taken from Box et al. (1978, Table 10.1, p. 308).

### 19.4.1 Estimates of effects and normal quantile plots

In the following example, y is a list of data from a 2-level factorial in standard order. The function `yates` returns the column of effects from Yates' algorithm.

```
> (def y (list 60 72 54 68 52 83 45 80)) ; standard order
Y
> (load "twolevel.lsp") ; file twolevel.lsp is loaded
; loading "twolevel.lsp"
T
> (yates y) ; Get final column in Yates' algorithm
(64.25 23 -5 1.5 1.5 10 0 0.5)
```

Delete the grand mean and define a variable `effects` to hold the estimated effects. The asterisk * stands for the previous expression (i.e., `(yates y)`).

```
> (def effects (cdr *))
EFFECTS
> effects
(23 -5 1.5 1.5 10 0 0.5)
```

To produce the normal probability plot of estimated effects, type

```
> (quantile-plot effects)
```

### 19.4.2 Identification of significant effects in unreplicated experiments

Without replications, it is usually difficult to identify significant effects unless these are very large. The standard method of judging the magnitude of an effect is the normal probability plot. This plot is however not invariant of the labels used for specifying the + and - levels of each factor. An invariant way of obtaining such a plot is via the `loh` function, which also superimposes a pair of boundary lines on the plot. Effects that fall outside the boundaries are considered significant. The necessary steps are as follows.

```
> (def y (list 60 72 54 68 52 83 45 80))
Y
> (load "twolevel")
```

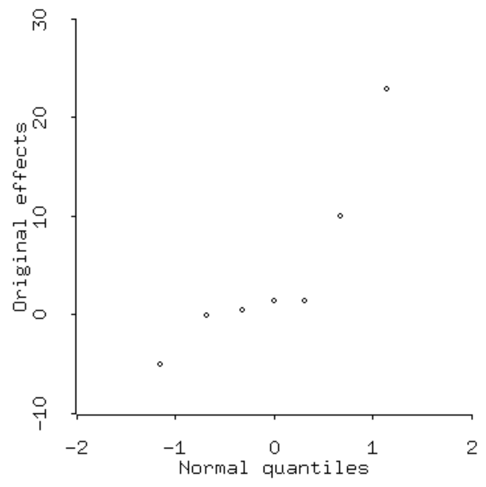

Figure 8: Plot of original effects

```
T
> (def effects (cdr (yates y)))
EFFECTS
> (loh effects)

Estimated effects without sign changes:
 (23.0 -5.0 1.5 1.5 10 0 0.5)
This version was updated on Jan 5, 2001.

Median minimizing transformation in standard order is:
 (-1 -1 1 -1 1 1 -1)
Ratio of slopes statistic =   2.456
90, 95, 97.5 and 99 percentiles of ratio of slopes are:
1.179, 1.516, 1.762 and 2.122.

An effect is possibly significant if its absolute value is > 10

Number of possibly significant effects = 1
```

Two plots are produced. One (Figure 8) is a normal quantile plot of the effects and another (Figure 9) is a normal quantile plot of sign-transformed effects. Horizontal lines in the second plot separate the significant from the insignificant effects. See Loh (1992) for details.

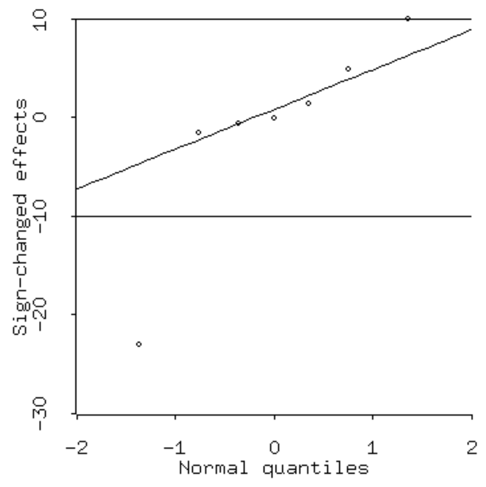

Figure 9: Plot of sign-transformed effects

### 19.4.3 Model checking

Suppose it is thought that only the three main effects and the 13 interaction are significant. The adequacy of this model can be checked with a plot of the residuals as follows. First get the estimated values $\hat{y}$ using the function `y-hat` with a list of the estimated effects (including the grand mean) with nonsignificant effects replaced by zeros. (This is similar to the reverse Yates' algorithm, except that the computer automatically does the inversion of the columns for you.)

```
> (y-hat (list 64.25 23 -5 0 0 10 0 0))
(60.25 73.25 55.25 68.25 50.25 83.25 45.25 78.25)
```

Next obtain the residuals by subtracting $\hat{y}$ from $y$.

```
> (def residuals (- y *))
> residuals (-0.25 -1.25 -1.25 -0.25 1.75 -0.25 -0.25 1.75)
```

Then plot the residuals to see if they fall on a straight line.

```
> (quantile-plot residuals)
```

### 19.4.4 Power transformations

Since normality of residuals and constant variance are two assumptions underlying the estimation techniques of factorial designs, transformations of the data should be considered if diagnostic plots suggest that these assumptions are severely violated in the original data. One method to achieve this is to raise the $y$ observations to some power $\lambda$. The command `transf` makes it easy to do this. First load the file `twolevel.lsp`. The following documentation is given by typing `(help 'transf)`:

```
Args: (y &key (response nil)(main-effects nil)(fraction nil)(last 3)
(first -2) (num 21))

Y is a list of observations from an unreplicated complete or
fractional 2-level factorial design, listed in standard order.  The Y
values must be POSITIVE numbers and the length of Y must be a power of
2.  This function produces 3 plots:

1. A plot of the residual sum of squares (RSS) of the transformed data
versus the power transformation lambda, assuming that the observations
are Gaussian with constant variance and that there are no interactions
(see Box, Hunter & Hunter, p. 335, Fig. 10.10).  The value of lambda
that minimizes the RSS is called the maximum likelihood estimator (MLE).

2. A plot of residuals versus predicted values, using a model that
contains only the significant effects (default).  If it is desired to
search for models containing only significant main effects but not
interactions, use the keyword :main-effects T. Significant effects are
determined using the method in Loh (1992), Identification of Active
Contrasts in Unreplicated Factorial Experiments.  COMPUTATIONAL
STATISTICS AND DATA ANALYSIS, vol. 14, pp. 135-148

3. A normal quantile plot of the residuals.

The value of lambda is controlled by a slider dialog box.  The initial
plots correspond to no transformation (lambda=1).

The default value for MAIN-EFFECTS is NIL, meaning that all
significant main effects and interactions are used in the models.  If
MAIN-EFFECTS is set to T, no interactions are allowed in the model.

The default value of NIL for FRACTION is for a complete factorial
design.  A non-nil value should be given for a fractional factorial.

For example, the command

(transf y)

fits a model containing all significant main effects and interactions, where
y is a list of yields from a complete factorial in standard order.
```

```
If the experiment is a complete factorial and only main effects are allowed
in the model, use the command

(transf y :main-effects t)

If the experiment is a fractional factorial with generators 4=12 and 5=123,
and only main effects are allowed in the model, use

(transf y :fraction (list '(1 2) '(1 2 3)) :main-effects t)

The default value of RESPONSE is T, i.e., the significant effects for
each lambda used in the transformation are printed in the command
window.

The values of lambda used in the slider dialog box are from a list
with smallest value FIRST and largest value LAST. The number of lambda
values (equally spaced) in this range is controlled by the parameter
NUM.  The default values are FIRST=-2, LAST=3, and NUM=21.

For example, to let lambda range over 11 values in (-0.5, 0.5), do

(transf y :first -0.5 :last 0.5 :num 11)
```

### 19.4.5 Analysis of replicated $2^k$ experiments

**Completely randomized experiment:** In this experiment, the runs are carried out in a completely randomized fashion. That is, some factor combinations may have more than one run completed before another factor combination has any.

The steps to analyze data from a replicated factorial is as follows.

1. Define the data in standard order, with replicated observations immediately following one another. For example, suppose the data are from the experiment in Table 1. The data should be defined as

   ```
   > (def y (list 22 23 33 32 32 23 38 32 41 36 54 64 18 26
     54 50 29 22 45 35 26 29 33 38 35 41 85 79 41 38 57 55))
   ```

2. To test for significance of the estimated effects at level $\alpha = 0.1$, use the command

   ```
   > (completely-randomized-replicates y 4 3.02)
   ```

   The syntax is `(completely-randomized-replicates y k modulus)`, where `k` is the number of factors and `modulus` is $|M|^{(\alpha)}_{g,\nu}$, the critical value from the *Studentized*

Table 1: Data for completely randomized replicated factorial experiment

| Factor levels | | | | |
|---|---|---|---|---|
| E | P | M | D | Yields |
| $-$ | $-$ | $-$ | $-$ | 22, 23 |
| $+$ | $-$ | $-$ | $-$ | 33, 32 |
| $-$ | $+$ | $-$ | $-$ | 32, 23 |
| $+$ | $+$ | $-$ | $-$ | 38, 32 |
| $-$ | $-$ | $+$ | $-$ | 41, 36 |
| $+$ | $-$ | $+$ | $-$ | 54, 64 |
| $-$ | $+$ | $+$ | $-$ | 18, 26 |
| $+$ | $+$ | $+$ | $-$ | 54, 50 |
| $-$ | $-$ | $-$ | $+$ | 29, 22 |
| $+$ | $-$ | $-$ | $+$ | 45, 35 |
| $-$ | $+$ | $-$ | $+$ | 26, 29 |
| $+$ | $+$ | $-$ | $+$ | 33, 38 |
| $-$ | $-$ | $+$ | $+$ | 35, 41 |
| $+$ | $-$ | $+$ | $+$ | 85, 79 |
| $-$ | $+$ | $+$ | $+$ | 41, 38 |
| $+$ | $+$ | $+$ | $+$ | 57, 55 |

*maximum modulus* distribution with $g = 2^k - 1$, $\nu = 2^k(m-1)$, and $m$ the number of replicates at each design point. Two plots will appear on the screen, a quantile plot of residuals and a plot of residuals versus fitted values, where the fitted values are computed from the significant effects determined from the simultaneous intervals. In addition the following lines will appear on the main screen:

```
Estimated effects in standard order are:
(39.5625 18.875 -5.375 -3.375 17.625 8.875 -6.625 -1.125
6.875 1.875 -1.375 -5.125 4.125 0.625 1.125 -4.125)
Estimate of sigma is: 4.301
Estimated SD of effect is: 1.521
Significant effects are:
(39.5625 18.875 -5.375 0 17.625 8.875 -6.625 0 6.875 0 0
-5.125 0 0 0 0)
Residuals are:
( -2.25 -1.25 3.875 2.875 11.625 2.625 2.5 -3.5 1.375
-3.625 -8.25 1.75 -4.5 3.5 -1.375 -5.375 3 -4
3.875 -6.125 -6.375 -3.375 -4.25  0.75 -6.375 -0.375
10.75 4.75 6.5 3.5 -0.125 -2.125)
```

The first line gives the grand mean followed by the Yates estimates in standard order. The second line gives the estimate of $\sigma$. The third gives the value of $\sqrt{4\hat{\sigma}^2/N}$. The

Table 2: A $2^3$ design in 3 replicates

| A | B | C | Trial 1 | Trial 2 | Trial 3 |
|---|---|---|---|---|---|
| $-$ | $-$ | $-$ | 13.35 | 14.22 | 11.58 |
| $+$ | $-$ | $-$ | 13.41 | 11.73 | 8.24 |
| $-$ | $+$ | $-$ | 14.32 | 17.70 | 17.16 |
| $+$ | $+$ | $-$ | 8.14 | 12.46 | 14.52 |
| $-$ | $-$ | $+$ | 16.97 | 17.48 | 16.65 |
| $+$ | $-$ | $+$ | 13.19 | 16.35 | 11.84 |
| $-$ | $+$ | $+$ | 17.73 | 18.78 | 15.93 |
| $+$ | $+$ | $+$ | 14.01 | 14.31 | 10.72 |

fourth line gives the effects in standard order with the non-significant effects replaced by zeroes. The residuals are given in the last line.

3. If you want to find a model based on the log transformation, use the command

```
> (completely-randomized-replicates (log y) 4 3.02)
```

To use a square root transformation, use

```
> (completely-randomized-replicates (ˆ y 0.5) 4 3.02)
```

Change the value of 0.5 in the command to another power if you wish. The original y values are not over-written since the transformation is done internally on the fly.

**Randomized complete blocks:** In this type of experiment, a complete factorial is performed one after another. Table 2 gives an example of three replicates of a complete $2^3$ experiment. The eight runs in the first trial are carried out in randomized order, then the eight runs in the second trial, etc. In order to eliminate block effects, Yates algorithm must be repeatedly applied to the data in each trial. The estimates are then averaged to give the final answers. The estimates of each effect from the independent trials are used to estimate the variance, which is needed in the simultaneous confidence intervals. The XLISPSTAT command is called `randomized-complete-blocks`. This is how it works if a test at level $\alpha = 0.1$ is desired.

```
> (def y1 (list 13.35 13.41 14.32 8.14 16.97 13.19 17.73 14.01))
> (def y2 (list 14.22 11.73 17.70 12.46 17.48 16.35 18.78 14.31))
> (def y3 (list 11.58 8.24 17.16 14.52 16.65 11.84 15.93 10.72))
> (randomized-complete-blocks (list y1 y2 y3) 2.72)

This version was revised on Jan 7, 2002.
```

```
Trial means: ( 13.89 15.3788  13.33)
Average effects in standard order (excluding mean):
(-3.57917 0.8975 -0.9975 2.26083 -0.274167 -1.06417 0.384167)
Estimated standard deviation of average effect: 0.7757
Simultaneous confidence interval half width: 2.110
Number of significant effects: 2
Significant effects in standard order (excluding mean):
(-3.57917 0 0 2.26083 0 0 0)
Residuals are:
((-1.19917 2.44 -0.229167 -2.83 0.16 -0.0408333 0.92 0.779167)
(-1.81792 -0.72875 1.66208 0.00125 -0.81875 1.63042 0.48125
-0.409583)
(-2.40917 -2.17 3.17083 4.11 0.4 -0.830833 -0.32 -1.95083))
```

The command requires two input arguments: a list of the yields from each trial (arranged in standard order) and $|M|_{g,\nu}^{(\alpha)}$, the critical value from the *Studentized maximum modulus* distribution, where $g = 2^k - 1$, $\nu = (m-1)(2^k - 1)$, and $m$ is the number of complete blocks. In this example, $|M|_{g,\nu}^{(\alpha)} = |M|_{7,14}^{(0.1)} = 2.72$. The command computes, for each replicate, the sample mean and the estimated effects. Since each effect is estimated multiple times, the estimates can be used to provide an estimate of their variance $\tau^2 = 4\sigma^2/2^k$. Averaging these estimates gives a pooled estimate of $\tau^2$, from which $\sigma^2$ can be estimated if necessary. As in the previous case, a quantile-plot of the residuals and a plot of residuals versus fitted values are produced.

To analyze the data in the log scale, use the command

```
> (randomized-complete-blocks (log (list y1 y2 y3)) 2.72)
```

Similarly, to analyze the data in the reciprocal scale, use

```
> (randomized-complete-blocks (ˆ (list y1 y2 y3)) 2.72)
```

### 19.4.6 Unequally replicated factorial designs

Suppose some of runs are replicated (possibly unequally). For example, suppose there is a replicate observation at the `(-,-,-)` set of conditions and two observations `60` and `61` were obtained instead of just `60`. The Yates' algorithm will not be able to handle this data, but the regression function will.

First enter the data in a file (called `ex1.lsp`, say) whose contents are:

```
-1 -1 -1 60
-1 -1 -1 61
```

```
 1 -1 -1 72
-1  1 -1 54
 1  1 -1 68
-1 -1  1 52
 1 -1  1 83
-1  1  1 45
 1  1  1 80
```

Read this file into XLISPSTAT as a matrix `z` and extract the column variables as `x1`, `x2`, `x3` and `y`:

```
> (def z (read-data-columns "ex1.lsp" 4))
> (def x1 (first z))
> (def x2 (second z))
> (def x3 (third z))
> (def y (fourth z))
```

Next define the interaction columns:

```
> (def x12 (* x1 x2))
> (def x13 (* x1 x3))
> (def x23 (* x2 x3))
> (def x123 (* x1 x2 x3))
```

Combine them into a matrix x and run the regression command, storing the results in a variable `result` (say):

```
> (def x (list x1 x2 x3 x12 x13 x23 x123))
> (def result (regression-model x y))

Least Squares Estimates:

Constant                   64.3125      (0.242061)
Variable 0                 11.4375      (0.242061)
Variable 1                -2.56250      (0.242061)
Variable 2                0.687500      (0.242061)
Variable 3                0.812500      (0.242061)
Variable 4                 5.06250      (0.242061)
Variable 5                 6.250000E-2  (0.242061)
Variable 6                0.187500      (0.242061)

R Squared:                0.999623
Sigma hat:                0.707107
Number of cases:                 9
Degrees of freedom:              1

RESULT
```

The estimated effects are twice the values given in the second column of the table, with estimated standard errors being twice those in parentheses. Variable 0 refers to the first component of `x`, variable 1 to the second component, etc. In this example, variables 5 and 6 (i.e., `x23` and `x123`) are not significant because their estimated values are less than twice their standard errors.

Notice that the output also gives the value of $\hat{\sigma}$ and the degrees of freedom.

The estimated main and interaction effects can be obtained and plotted by doing:

```
> (def effects (* 2 (cdr (send result :coef-estimates))))
EFFECTS
> effects
(22.875 -5.125 1.375 1.625 10.125 0.125 0.375)
> (quantile-plot effects)
```

The residuals and predicted values $\hat{y}$ are gotten by:

```
> (def residuals (send result :residuals))
RESIDUALS
> residuals
(-0.5 0.5 0 0 0 0 0 0 0)
> (send result :fit-values)
(60.5 60.5 80 72 54 68 52 83 45)
```

You get a normal probability plot of the residuals by:

```
> (quantile-plot residuals)
```

and a plot of residuals versus predicted values by:

```
> (send result :plot-residuals)
```

*Notes:*

1. It is not necessary for the data in the file `ex1.lsp` to be in standard order, even if the design is unreplicated.

2. Data from a fractional factorial design can be analyzed similarly.

## 20 XLISPSTAT archives

The following archives contain more user-contributed XLISPSTAT functions.

- UCLA XLISPSTAT Archive: `http://www.xlispstat.org/`
- CMU Statlib XLISPSTAT Archive: `http://lib.stat.cmu.edu/xlispstat/`
- Penn State XLISPSTAT Projects: `http://euler.bd.psu.edu/lispstat`

## References

G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters*. Wiley, New York, 1978.

W.-Y. Loh. Identification of active contrasts in unreplicated factorial experiments. *Computational Statistics and Data Analysis*, 14:135–148, 1992.

L. Tierney. *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. Wiley, New York, 1990.