WEIGHTS

PURPOSE

Specifies the weights for subsequent FIT commands.

DESCRIPTION

If no weights are specified via the WEIGHTS command, the FIT command defaults to equal weights. The WEIGHTS command is used to incorporate additional information (namely, the nature of the error structure) into the fitting process. The net result is improved least squares fitting estimates. The desired weight values must be placed in a variable by the analyst. This placement is most commonly done via the SERIAL READ, the READ, or the LET commands. For example, to specify that 25, 25, 50, 80, and 70.5 are to be the weight values in a linear fit of Y on X, one could enter the following sequence of commands:

```
LET X2 = DATA 25 25 50 80 70.5
WEIGHTS X2
FIT Y = A+B*X
```

Weighted fits are typically used in the following two cases:

1. When the analyst has pre-specified weights;
2. To perform various types of robust fitting.

For case 1, the weights are values which reflect the analyst's confidence in the precision of the response value. High-precision response values are assigned high weights, middle-precision response values are assigned middle weights, and low-precision response values are assigned low weights.

Least squares fitting generates optimal fits given independent normally distributed errors with constant variance. However, if the errors do not satisfy these assumptions, then the least squares estimates may be unduly affected. Robust fitting methods attempt to generate good fits for a range of error distributions. In particular, they protect against heavy tailed distributions which have outliers relative to a normal distribution. Robust estimates should be stable if a few values are changed significantly or if a large number of values are changed by a small amount. One class of robust methods works by down-weighting large residuals. This approach is discussed in more detail in the NOTE sections below.

SYNTAX

WEIGHTS <x>
where <x> specifies the name of a pre-existing variable which contains the desired weights.

EXAMPLES

```
WEIGHTS W
```

NOTE 1

One class of robust techniques, called M-estimators, are maximum likelihood estimators that minimize the sum of a function of the residuals. M-estimators can typically be approximated through a technique called iteratively reweighted least squares (or IRLS). The basic algorithm is to start with a least squares fit, weight the residuals by some formula, and then apply a weighted fit. The new residuals are then weighted by the same formula. This continues until the residuals reach some type of convergence criterion. The Heiberger and Becker article (see the REFERENCE section below) describe a sophisticated function for performing this process with the S statistical package. However, the basic steps can be applied to any program that supports weighted least squares.

There is no single weighting scheme that works best in all cases. Various weighting schemes have been proposed and used. The Heiberger and Becker article gives 10 of the more popular ones.

The program examples below demonstrate IRLS using DATAPLOT. The second program example is a macro (called IRLS.DP) that performs the IRLS. The program 1 example is a driver that specifies the fit to use and the desired weighting function. The IRLS.DP macro can be modified in a straightforward manner:

1. Additional weighting schemes can be added (or ones that you don't use can be deleted).
2. The maximum number of iterations and the convergence criterion can be changed to suit your preference.

NOTE 2

There are other approaches to robust estimation. R-estimators are based on ranks. L-estimators are linear functions of the sample order statistics (this includes least median squares regression and quantile regression). DATAPLOT does not support these approaches to robust estimation at this time.

NOTE 3

Least absolute deviations (LAD) is another robust method. This method minimizes:

ABSOLUTE VALUE(Y - YPRED)

Although LAD is usually estimated with linear programming techniques, it can also be estimated with the IRLS algorithm. The program 2 example uses a method of Tukey (see the Mosteller and Tukey book in the REFERENCE section below). This method gives small residuals a weight of 1 (otherwise small residuals tend to have unduly large weights).

NOTE 4

Robust methods based on weighting the residuals provide protection against outliers in the dependent variable. However, it does not necessarily protect against outliers in the independent variables. Points with high leverage (see the documentation for the FIT command) can exert enough influence on the fit that they will have small residuals even though they might be outliers. The Hamilton book (see the REFERENCE section below) offers the following suggestion for this problem:

1. Let h(i) be the leverage for the ith case. The leverage values are written to a file for linear and multi-linear fits (see the documentation for the FIT command for details). Let CH be the 90th percentile of the h(i). In DATPLOT, do:

   ```
   FIT ...
   SKIP 1
   READ DPST3F.DAT JUNK HI
   LET CH = NINTH DECILE HI
   ```

2. Calculate a weight based on leverage as follows (assume the HI and CH were calculated as in step 1):

   ```
   LET WEIGHTH = HI
   LET WEIGHTH = 1 SUBSET HI <= CH
   LET WEIGHTH = (CH/HI)**2 SUBSET HI > CH
   ```

3. In the IRLS algorithm, multiply the robustness weights by WEIGHTH. The WEIGHTH values are only computed once, but they are multiplied by the robustness weights at each iteration in the IRLS.

NOTE 5

Robust techniques generally involve a trade-off between efficiency and robustness. Efficiency refers to how well the method performs when the errors do in fact satisfy the least squares assumptions. Robustness refers to the performance over a range of error distributions. The trade-off is usually handled by specifying a tuning constant. Most of the tuning constants in the program 2 example are chosen to yield 95% efficiency.

NOTE 6

The LET subcommands BIWEIGHT and TRICUBE provide built-in weighting functions. See the documentation for these commands for details.

NOTE 7

Weighted fits can be applied to both linear and non-linear fits. The LOWESS command automatically performs biweight weighting. The other smooth and fit commands in DATAPLOT (PRE-FIT, SPLINE FIT, SMOOTH) do not support weighted fitting at this time, so the IRLS algorithm cannot be used.

DEFAULT

None (i.e., all values are assigned equal weights).

SYNONYMS

None

RELATED COMMANDS

FIT	=	Perform weighted (or unweighted) linear or non-linear fits.
BIWEIGHT	=	Perform robust fit via biweight residuals.
TRICUBE	=	Perform robust fit via tricube residuals.

REFERENCES

"Applied Linear Statistical Models," 3rd ed., Neter, Wasserman, and Kunter, 1990, Irwin (provides the details of weighted least squares).

"Data Analysis and Regression," Mosteller and Tukey, Addison-Wesley, 1977 (chapter 14).

"Regression With Graphics," Hamilton, Wadsworth, 1992.

"Design of an S Function for Robust Regression Using Iteratively Reweighted Least Squares," Heiberger and Becker, Journal of Computational and Graphical Statistics, September, 1992.

APPLICATIONS

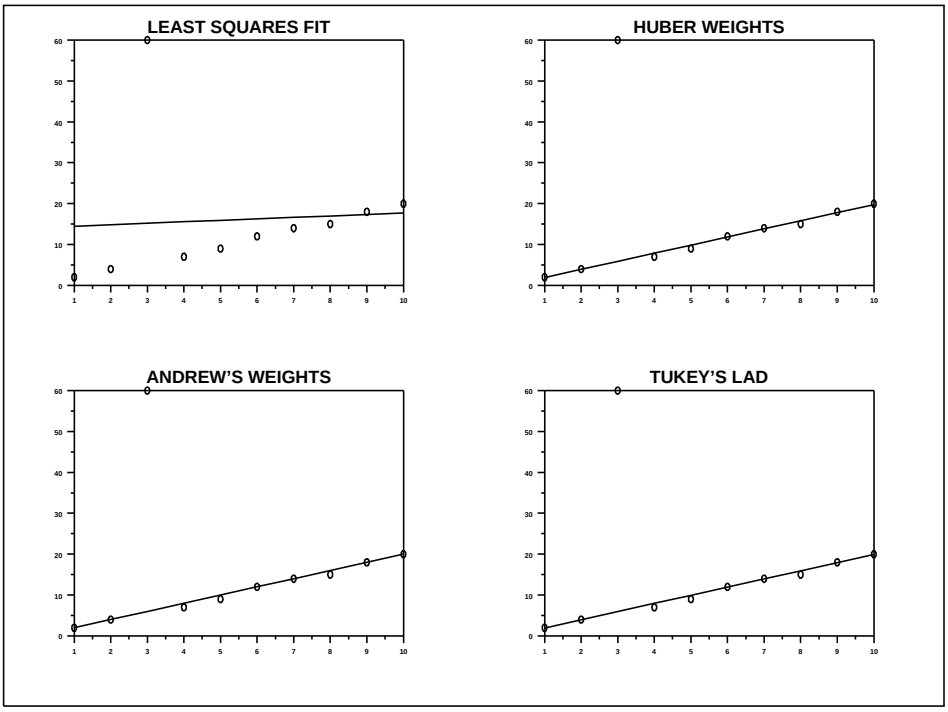
Robust fitting

IMPLEMENTATION DATE

87/9

PROGRAM 1

```
LET X = DATA 1 2 3 4 5 6 7 8 9 10
LET Y = DATA 2 4 60 7 9 12 14 15 18 20
LET STRING F = FIT Y X
CHARACTER CIRCLE BLANK; LINE BLANK SOLID
MULTIPLOT 2 2; MULTIPLOT CORNER COORDINATES 0 0 100 100
FIT Y X
TITLE LEAST SQUARES FIT
PLOT Y PRED VS X
LET METHOD = 5
CALL IRLS.DP
TITLE HUBER WEIGHTS
PLOT Y PRED VS X
LET METHOD = 1
CALL IRLS.DP
TITLE ANDREW'S WEIGHTS
PLOT Y PRED VS X
LET METHOD = 11
CALL IRLS.DP
TITLE TUKEY's LAD
PLOT Y PRED VS X
END OF MULTIPLOT
```



PROGRAM 2 (the IRLS.DP macro used by program example 1)

```
FEEDBACK OFF
. Compute iteratively re-weighted least squares in DATAPLOT.
. 1) The parameter METHOD should be defined before caalling this
.    macro.  It can have one of the following values:
.       1   = Andrew's method
.       2   = Tukey's bisquare
.       3   = Cauchy
.       4   = Hampel
.       5   = Huber
.       6   = Logistic
.       7   = median
.       8   = Welsch
.       9   = fair
.      10   = Talworth
.      11   = Tukey's approximation for least absolute deviation
.     You can modify this macro to include other methods as well.
.     In addition, you can modify the tuning constant.  Most of
.     these are chosen so that the method is 95% efficient for
.     error terms that are in fact normally distributed.
. 2) The following assumes that a string F has been defined
.     before calling this macro to define the type of fit.  E.g.,
.          LET STRING F = FIT Y X
. 3)  The convergence criterion and the maximum number of
.     iterations can be modified.
. 4) The following algorithm is used:
.     a) Perform an initial unweighted least squares estimate
.     b) Scale the residuals:
.          Ui = Ei/s
.       where s = Median Absolute Deviation/0.6745
.     c) Apply the specified weight function
.     d) Check for convergence
.
WEIGHT
^F
LET MAXITER = 10
LOOP FOR K = 1 1 MAXITER
       LET RESOLD = RES; LET MED = MEDIAN RES
      LET TEMP = ABS(RES - MED);  LET MAD = MEDIAN TEMP
      LET S = MAD/0.6745;  LET U = RES/S
      IF METHOD = 1
                LET C = 1.339
                LET TAG = ABS(U/C)
                LET WT = 1 SUBSET U = 0
                LET WT = 0 SUBSET TAG > PI
                LET WT = SIN(TAG)/TAG  SUBSET TAG <= PI
      END OF IF
      IF METHOD = 2
                LET C = 4.685
                LET TAG = ABS(U/C)
                LET WT = 0 SUBSET TAG > 1
                LET WT = (1 - TAG**2)**2  SUBSET TAG <= 1
      END OF IF
      IF METHOD = 3
                LET C = 2.385
                LET WT = 1/(1 + (u/c)**2)
      END OF IF
```

```
    IF METHOD = 4
            LET C = 8; LET A = 2; LET B = 4
            LET TAG = ABS(U)
            LET WT = 1 SUBSET U <= A
            LET WT = A/TAG  SUBSET TAG > A   SUBSET TAG <= B
            LET WT = (A/TAG)*((C-TAG)/(C-B)) SUBSET TAG > B SUBSET TAG <= C
            LET WT = 0 SUBSET U > C
    END OF IF
    IF METHOD = 5
            LET C = 1.345
            LET TAG = ABS(U)
            LET WT = 1 SUBSET TAG <= C
            LET WT = C/TAG  SUBSET TAG > C
    END OF IF
    IF METHOD = 6
            LET C = 1.205
            LET TAG = ABS(U/C)
            LET WT = 1 SUBSET U = 0
            LET WT = TANH(TAG)/TAG  SUBSET TAG <> 0
    END OF IF
    IF METHOD = 7
            LET TAG = ABS(U)
            LET WT = 1/0.000001 SUBSET U = 0
            LET WT = 1/TAG SUBSET U <> 0
    END OF IF
    IF METHOD = 8
            LET C = 2.985
            LET TAG = (U/C)**2
            LET WT = EXP(-0.5*TAG)
    END OF IF
    IF METHOD = 9
            LET C = 1.4
            LET WT = 1/(1 + (u/c))**2
    END OF IF
    IF METHOD = 10
            LET C = 2.795
            LET TAG = ABS(U)
            LET WT = 1 SUBSET TAG <= C
            LET WT = 0 SUBSET TAG > C
    END OF IF
    IF METHOD = 11
            LET TEMP = ABS(RES)
            LET C = MEDIAN TEMP
            LET TAG = ABS(Y - PRED)
            LET WT = C/TAG  SUBSET TAG > C
            LET WT = 1 SUBSET TAG <= C
    END OF IF
    WEIGHTS WT
    ^F
    LET DELTA = (RESOLD - RES)**2
    LET NUM = SUM DELTA; LET NUM = SQRT(NUM)
    LET DELTA2 = RESOLD*RESOLD
    LET DENOM = SUM DELTA2; LET CONV = NUM/DENOM
    IF CONV <= 0.0001
            BREAK LOOP
    END OF IF
END OF LOOP
```