Graphs in the LINGO Modeling Language

LINDO Systems

www.lindo.com

LINGO is a general purpose algebraic language for describing and solving optimization problems.

It also has extensive graphing/charting capabilities for describing the results of your analysis.

Graphs in LINGO

Both your audience and you can more quickly understand the results of your optimization analysis if you use graphs/charts.

LINGO provides over a dozen different graph types to help you easily describe the results of your optimization analysis.

Here are examples of the half dozen more popular types. In LINGO, graphs/plots/charts are called “charts” for short.

Simple calls to all chart types are illustrated in the LINGO directory in:

Samples¥Charts.lg4

Graphs in LINGO

In the following we illustrate:

+ Function plots of one variable: **@CHARTPCURVE**
+ Function and contour plots of two variables: **@CHARTPSURFACE** , **@CHARTPCONTOUR**
+ Histograms: **@CHARTHISTO**
+ Bar charts: **@CHARTBAR**
+ Networks: **@CHARTNETNODE**
+ Space-Time diagrams: **@CHARTSPACETIME**
+ Trade-off curves, tabulated: **@CHARTCURVE**
+ Tornado diagrams: **@CHARTTORNADO**
+ Scatter charts: **@CHARTSCATTER**

All the examples here can be found in the MODELS library at **www.lindo.com**.

Click on: **HELP -> MODELS Library -> Alphabetical index.**

Which Chart Types are Available?

In LINGO Click on:

`Edit -> Paste Function -> Charting`

Charting >	@CHARTBAR('title', 'x-axis', 'y-axis', 'legend1', x1[, ..., 'legendn', xn]);
Date, Time and Calendar >	@CHARTBUBBLE('title', 'x-axis', 'y-axis', 'leg1', x1, y1, size1[, ..., 'legendn', xn, yn, sizen]);
Distributions >	@CHARTCONTOUR('title', 'x-axis', 'y-axis', 'legend', x, y, z);
External Files >	@CHARTPCONTOUR('title', 'x-axis', 'y-axis', 'z-axis', proc, x, xl, xu, y, yl, yu, 'legend', z);
Financial >	@CHARTCURVE('title', 'x-axis', 'y-axis', 'legend1', x1, y1[, ..., 'legendn', xn, yn]);
Mathematical >	@CHARTPCURVE('title', 'x-axis', 'y-axis', proc, x, xl, xu, 'legend1', y1[, ..., 'legendn', yn]);
Matrix >	@CHARTHISTO('title', 'x-axis', 'y-axis', 'legend', bins, x);
Probability >	@CHARTLINE('title', 'x-axis', 'y-axis', 'legend1', x1[, ..., 'legendn', xn]);
Programming >	@CHARTNETARC('title', 'x-axis', 'y-axis', 'legend1', x1, y1, x2, y2[, ..., 'legendn', x1n, y1n, x2n, y2n]);
Report >	@CHARTNETNODE('title', 'x-axis', 'y-axis', 'legend1', x1, y1, i1, j1[, ..., 'legendn', xn, yn, in, jn]);
Set Handling >	@CHARTPIE('title', 'legend', x);
Set Looping >	@CHARTRADAR('title', 'legend1', x1[, ..., 'legendn', xn]);
Stochastic Programming >	@CHARTSCATTER('title', 'x-axis', 'y-axis', 'legend1', x1, y1[, ..., 'legendn', xn, yn]);
Trigonometric >	@CHARTSPACETIME('title', 'x-axis', 'y-axis', 'legend1', x1, y1, x2, y2[, ..., 'legendn', x1n, y1n, x2n, y2n]);
Variable Domain >	@CHARTSURFACE('title', 'x-axis', 'y-axis', 'z-axis', 'legend', x, y, z);
Other >	@CHARTPSURFACE('title', 'x-axis', 'y-axis', 'z-axis', proc, x, xl, xu, y, yl, yu, 'legend', z);
	@CHARTTORNADO('title', 'x-axis', 'y-axis', base, 'legendh', high, 'legendl', low);

Plots of Functions in LINGO

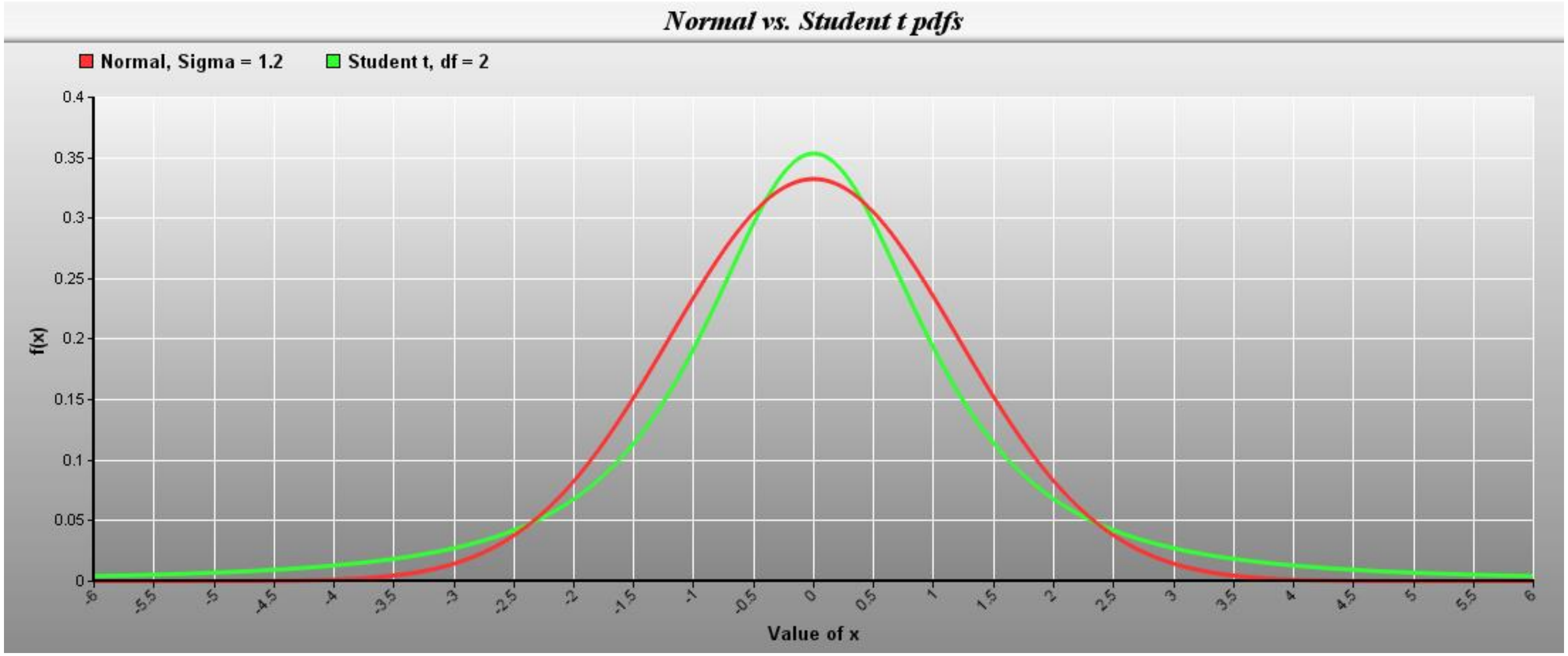
In finance, the "Student *t* " distribution, because it has "fatter tails" than the Normal, is sometimes recommended as better than the Normal as a model of the randomness of financial returns from a stock investment. A graph or chart is a good way of showing the difference.

The following little LINGO program illustrates:

a) how to generate a chart/plot/graph of a function,
b) the wide range of probability distributions available in LINGO,
c) how to use a procedure or function to record a computation that will be used repeatedly.

Graphs and Procedures in LINGO

This graphs suggests that
a “fat tailed” distribution could also be called a “peaked” distribution.



Univariate functions can be graphed using the
`@chartpcurve( )` routine.

Here is the LINGO Code that Generates the Plot

```
! Plotting using Procedures,  (PlotNormalCurve.lng)
PROCEDURE CALCPDF:
! A procedure that calculates the probability density function
  for the Normal and for the Student t distributions;
  YN = @PNORMPDF( MU, SIGMA, X);
  YT = @PSTUTPDF( DF, X);
ENDPROCEDURE

CALC:
  @SET( 'TERSEO',2); ! Output level (0:verb, 1:terse, 2:only errors);
  DF = 2;  ! Degrees of freedom for the t distribution.
             As DF approaches infinity, the t approaches the Normal;
  MU = 0;  ! Mean of the Normal;
  SIGMA = 1 + 0.5/DF; ! S.D. of a similar Normal;
  @CHARTPCURVE('Normal vs. Student t pdfs',
     'Value of x',   ! Label of X axis;
     'f(x)',         ! Label of Y axis;
     CALCPDF,        !Function to compute point on graph;
     X, -5*SIGMA, 5*SIGMA, ! X axis variable, bounds;
    'Normal, Sigma = '+@FORMAT(SIGMA,'5.1f'), YN,  ! Curve 1;
    'Student t, df = '+@FORMAT(DF,'5.1f'),    YT); ! Curve 2;
ENDCALC
```

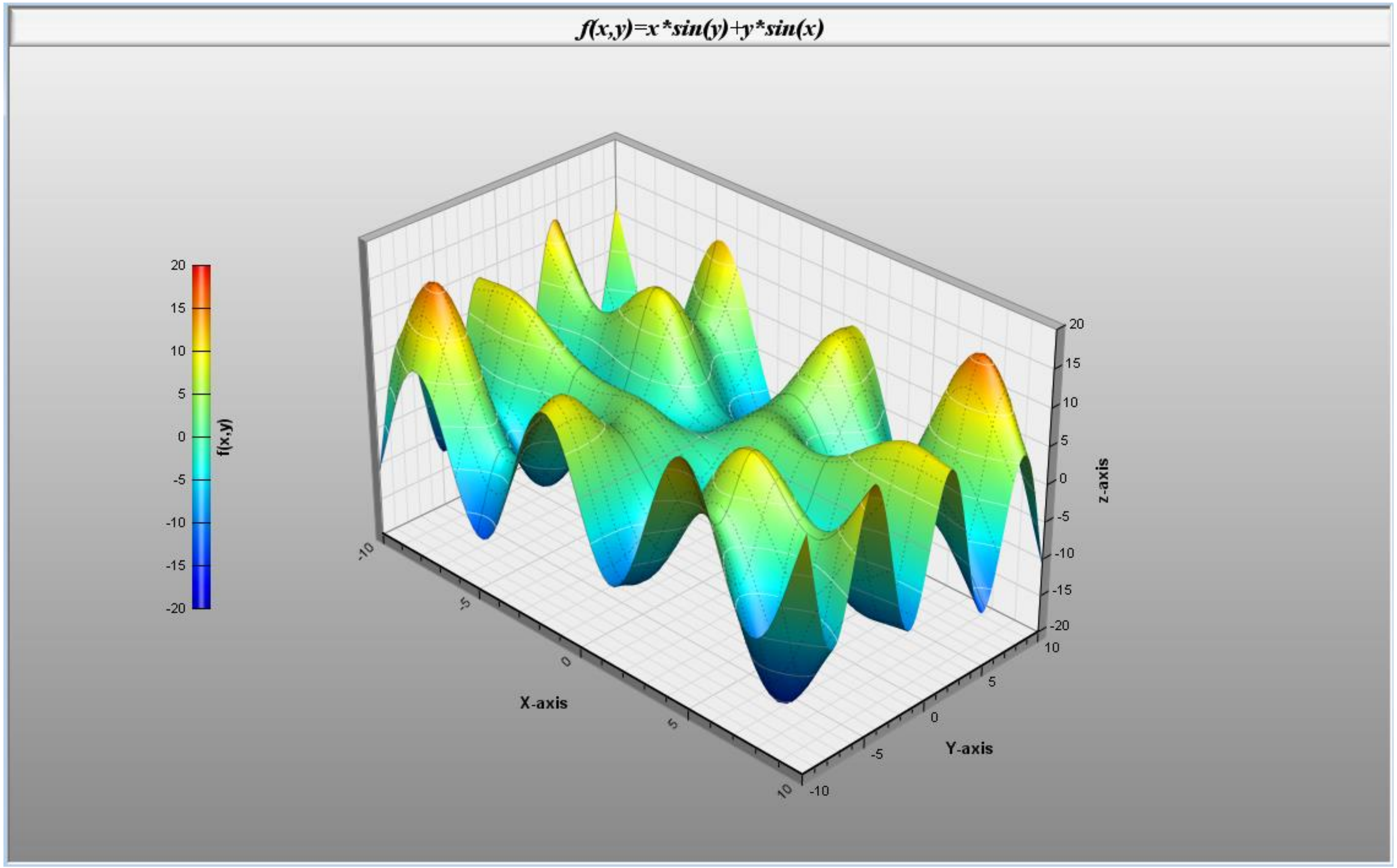
You can change the degrees of freedom, DF, to see how the distribution shape changes.

Graphs/Charts of Functions of Two Variables

With the use of color or shading, one can also illustrate the features of a function of two variables.

Graphs of Functions of Two Variables

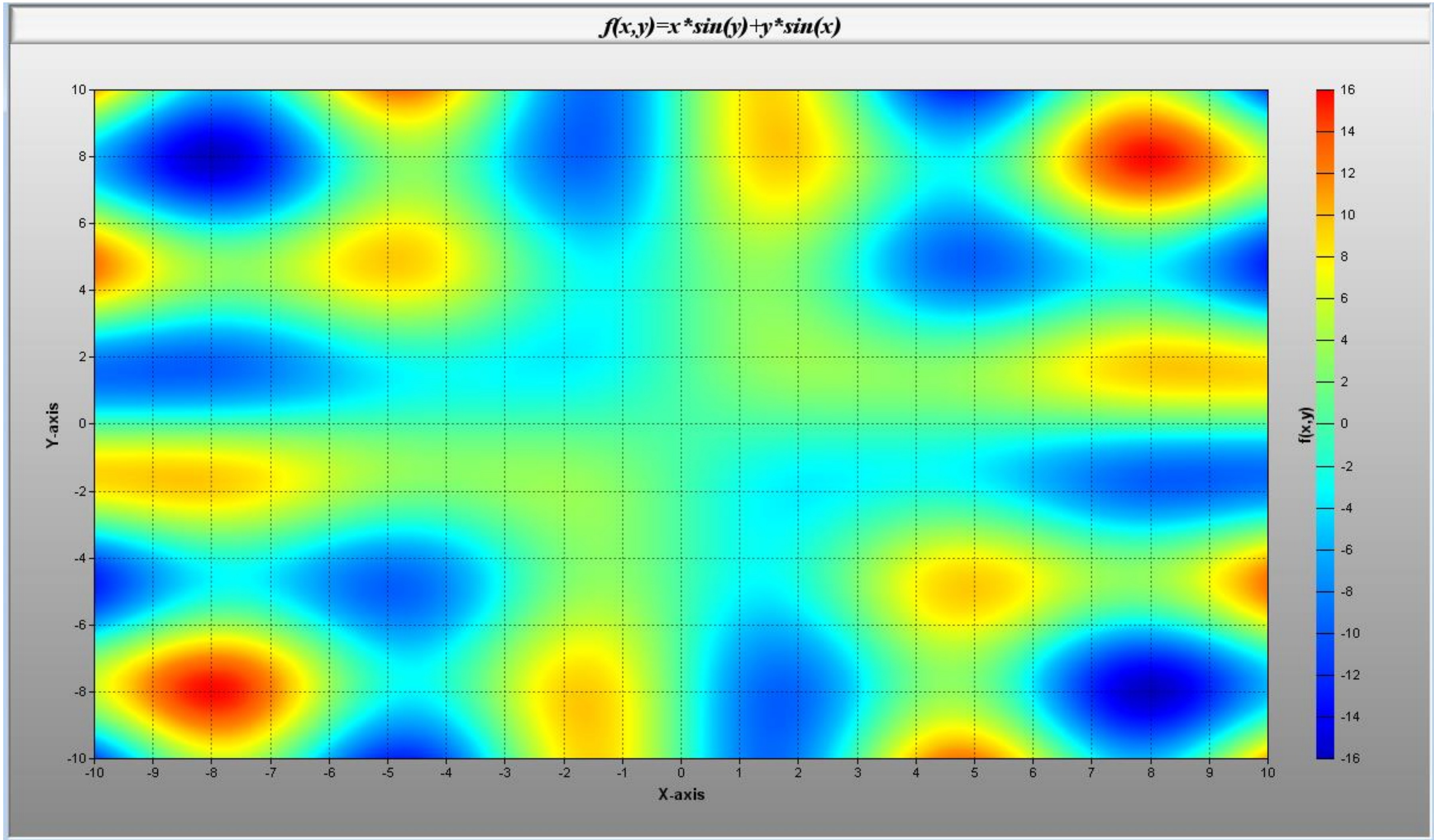
This is a “Surface Map” generated with @CHARTPSURFACE( ).



Where do you think the global minimum occurs?

Graphs of Functions of Two Variables, Contour Map

A Contour or “Heat” map is more suggestive of where the minimum occurs.



Graphs of Function of Two Variables, the Code

```
! Given a function of 2 variables,    (ChartPcontour.lng)
  1) Create Surface and Contour charts of it,
  2) Find a global minimum ;
PROCEDURE bivariatefun:
  fval = x*@sin(y) + y*@sin(x);
ENDPROCEDURE

SUBMODEL bivariateopt:
  min = fxy;
  fxy = x*@sin(y) + y*@sin(x);
  @free( fxy);  ! It could be < 0;
  @bnd( lb, x, ub);
  @bnd( lb, y, ub);
ENDSUBMODEL
```

Graphs of Function of Two Variables, the Code

```
CALC:
  LB = -10; ! Lower bound for x & y;
  UB =  10; ! Upper bound for x & y;

 @CHARTPSURFACE( 'f(x,y)=x*sin(y)+y*sin(x)', ! Chart title;
      'X-axis', 'Y-axis', 'z-axis', ! Titles of the axes;
      bivariatefun,      ! Name of function procedure;
      x, lb, ub,         ! X variable and bounds;
      y, lb, ub,         ! Y variable and bounds;
      'f(x,y)', fval); ! Output title and variable;

 @CHARTPCONTOUR( 'f(x,y)=x*sin(y)+y*sin(x)', ! Chart title;
      'X-axis', 'Y-axis', 'z-axis', ! Titles of the axes;
      bivariatefun,      ! Name of function procedure;
      x, lb, ub,         ! X variable and bounds;
      y, lb, ub,         ! Y variable and bounds;
      'f(x,y)', fval); ! Output title and variable;

! Optimize it;
  @SET( 'GLOBAL',1);     ! Use Global solver? 0:No, 1:Yes;
  @SOLVE( bivariateopt);
  @WRITE(' f(x,y) achieves global minimum value = ', fxy,
           ', at x = ',x,', y = ',y, @NEWLINE(1));
ENDCALC
```

```
f(x,y) achieves global minimum value = -15.8334547,
       at x = -7.9786657, y = 7.9786657.
```

Histograms of Statistical Data

Histograms are a good way for quickly understanding the variability of some unpredictable quantity such as demand, or profit as a function of a number of random parameters.

We illustrate histograms in LINGO with the following simple process.

Take a sample of *NS* observations,
Each observation containing *NU* uniform random numbers.

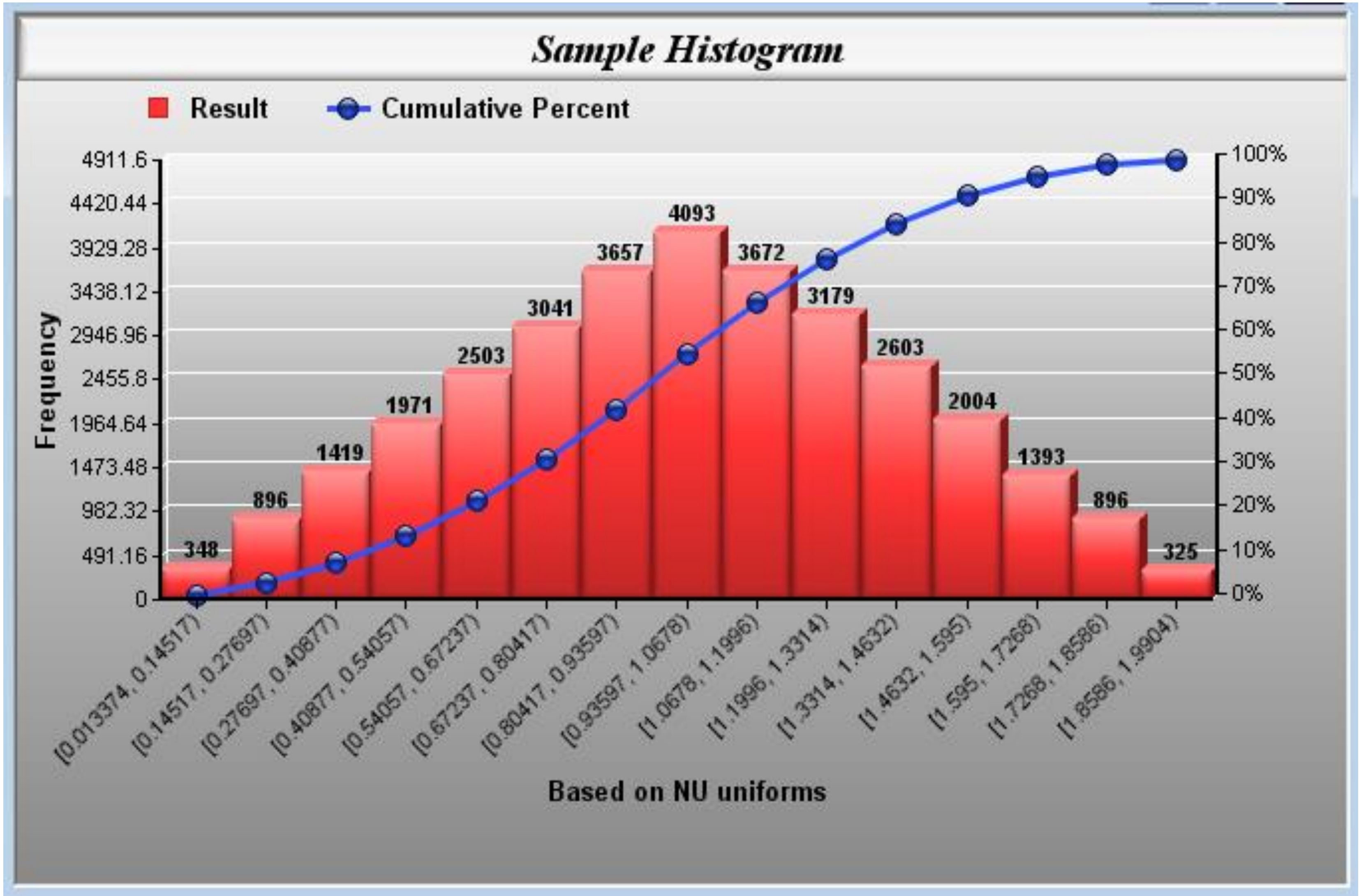
Combine each set of *NU* numbers into a single number using either sum, min, max, or product,
and plot the histogram using `@CHARTHISTO( );`

Can you predict the shape of the distribution for each case?
E.g., What does the distribution look like for: Sum of 2 uniforms?
Minimum of 2 uniforms?

Histograms, - Limiting Distributions

```
! Illustrate Histogram feature of LINGO.  (CentralLimitTheorem.lng)
SETS:
  OBS: RESULT;
  RV: DRAW;
ENDSETS
CALC:
  NS = 32000; ! Number observations;
  NU = 2;     ! Number uniform random numbers in each obs.;
  UI = 0.43928; ! An initial random number seed;
  NBINS = 15; ! Optionally specify number bins;
  OBS = 1..NS; ! Declare set of observations;
  RV = 1..NU;  ! Declare set of vars. per obs.;
   @FOR( OBS( i):
      @FOR( RV(j):
           UI = @RAND( UI); ! Get another uniform;
        DRAW(j) =  UI;
        );
      RESULT(i) = @SUM( RV(j): DRAW(j));
  !    RESULT(I) = @MIN( RV(j): DRAW(j));
  !    RESULT(I) = @MAX( RV(j): DRAW(j));
  !    RESULT(I) = @PROD( RV(j): DRAW(j));
        );
  ! A histogram with NBINS bins;
  ( 'Sample Histogram', ! Title;
    'Based on NU uniforms', ! X axis label;
    'Frequency',            ! Y axis label;
    'Result',  ! Label for data set plotted;
     NBINS, RESULT);
  ! If you choose NBINS = 0, then @CHARTHISTO will choose;
ENDCALC
```

Histogram Output



Can you predict distribution shape for different NU, or using MIN, MAX or PROD instead of SUM?

Bar Charts & Staff Scheduling

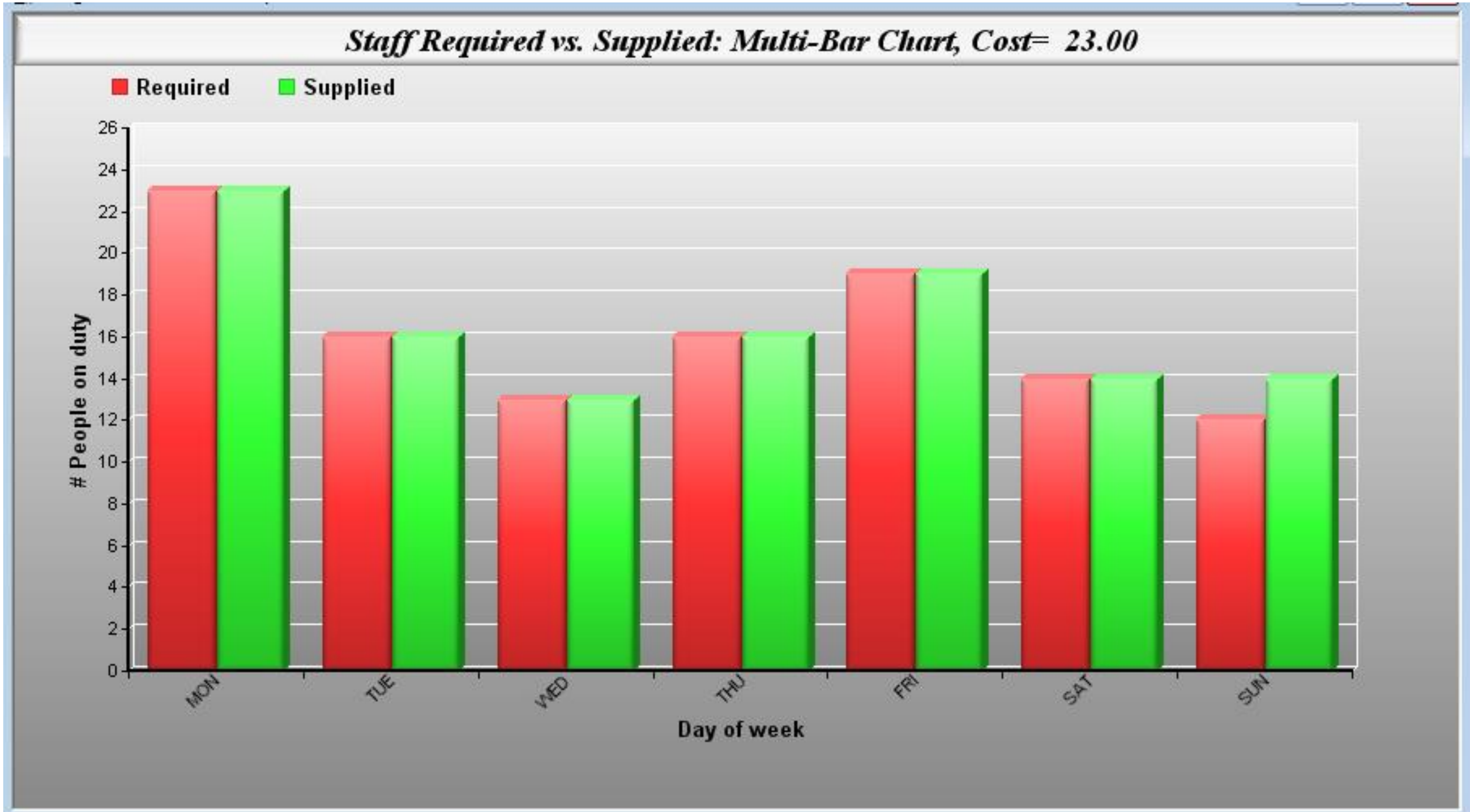
Bar charts are useful for displaying the relative levels of various activities.

```
! The simplest possible work staffing model in LINGO  (StaffPatSimp.lng);
SETS:
 ! The generic data structures;
   PERIOD:  REQUIRED, SUPPLIED;
   WORKPAT: NUM2RUN, COST;
   WXD( WORKPAT, PERIOD): ONRNOT;
ENDSETS

DATA:
 ! The circular week staffing problem;
 ! The names for the periods;
   PERIOD   = MON   TUE   WED   THU   FRI   SAT   SUN;
 ! Number folks required on duty each period;
   REQUIRED= 19    17    15    19    17    14    12;
 ! The possible work patterns. ONRNOT(i,j) = 1
   if workpattern i has someone on duty in period j;
   ONRNOT =    1     1     1     1     1     0     0 ! Start MON;
               0     1     1     1     1     1     0 ! Start TUE;
               0     0     1     1     1     1     1
               1     0     0     1     1     1     1
               1     1     0     0     1     1     1
               1     1     1     0     0     1     1
               1     1     1     1     0     0     1 ! Start SUN;
               ;
! Cost per unit of each work pattern;
     COST =    1     1     1     1     1     1     1;
!  Make SAT and SUN slightly less attractive;
!     COST =    1 1.01 1.05 1.05 1.05 1.05 1.04;
 ENDDATA
```

LINDO SYSTEMS INC.

Bar Charts: All Days of Week Are Equally Costly



We have extra staff on Sunday. Some people prefer to watch football rather than work on Sunday.

Bar Charts: We Make SAT and SUN Slightly More Costly

We no longer over-staff on Sunday…

How do we know that for this data set,

19 17 15 19 17 14 12

*we will over-staff by at least 2 person*days?*

LINDO SYSTEMS INC.

Bar Charts & Staffing: The Code for the Model

```
SUBMODEL STAFFIT:
! Minimize cost of people used over all patterns;
MIN = OBJ;
  OBJ= @SUM( WORKPAT( i): COST(i)*NUM2RUN( i));

! For each period, the patterns used must cover that period;
@FOR( PERIOD( j):
  SUPPLIED( j) = @SUM( WORKPAT( i): ONRNOT(i,j)*NUM2RUN(i));
  SUPPLIED( j) >= REQUIRED( j); ! Meet demand;
  );

! Can only hire integer people;
 @FOR(WORKPAT(j): @GIN(NUM2RUN(j)));

ENDSUBMODEL
```

Bar Charts & Staffing: Code to Generate the Chart

```
CALC:
  @SOLVE( STAFFIT);

 ! Write a lil report;
  @FOR( WORKPAT(i) | NUM2RUN #GT# 0:
    @WRITE(' Put ', NUM2RUN(i),' people on the work pattern:',
@NEWLINE(1));
      @FOR(PERIOD(j): @WRITE('    ',ONRNOT(i,j)));
    @WRITE( @NEWLINE(2));
      );
! Generate a bar chart of Required vs. Supplied;
   @CHARTBAR('Staff Required vs. Supplied: Multi-Bar Chart, Cost= `
     +@FORMAT(OBJ,'6.1f'),! Title of chart;
   'Day of week',           ! X axis label;
   '# People on duty',      ! Y axis label;
   'Required', REQUIRED,    ! Quantity 1;
   'Supplied', SUPPLIED);   ! Quantity 2;
ENDCALC
```

Network and Routing Problems and Their Graphs

When you are trying to decide what is the best way of getting from one place to another, a map with the proposed route traced on it is usually helpful.

Frequently, optimization is useful for deciding the best/shortest route.

The following illustrate:

1) Solution of a Traveling Salesman Problem (only how to use, not the details of the model)
2) Generate a network chart.

We start on a sad musical note…

The Day the Music Died

January 23rd, 1959, Some “up-and-coming” singers from Texas, Buddy Holly, Ritchie Valens, J.P. "The Big Bopper" Richardson, Waylon Jennings, Dion DiMucci (of the Belmonts), start the “Winter Dance Party” tour.

24 Rock-and-Roll concerts in 24 different U.S. cities in 24 different days!

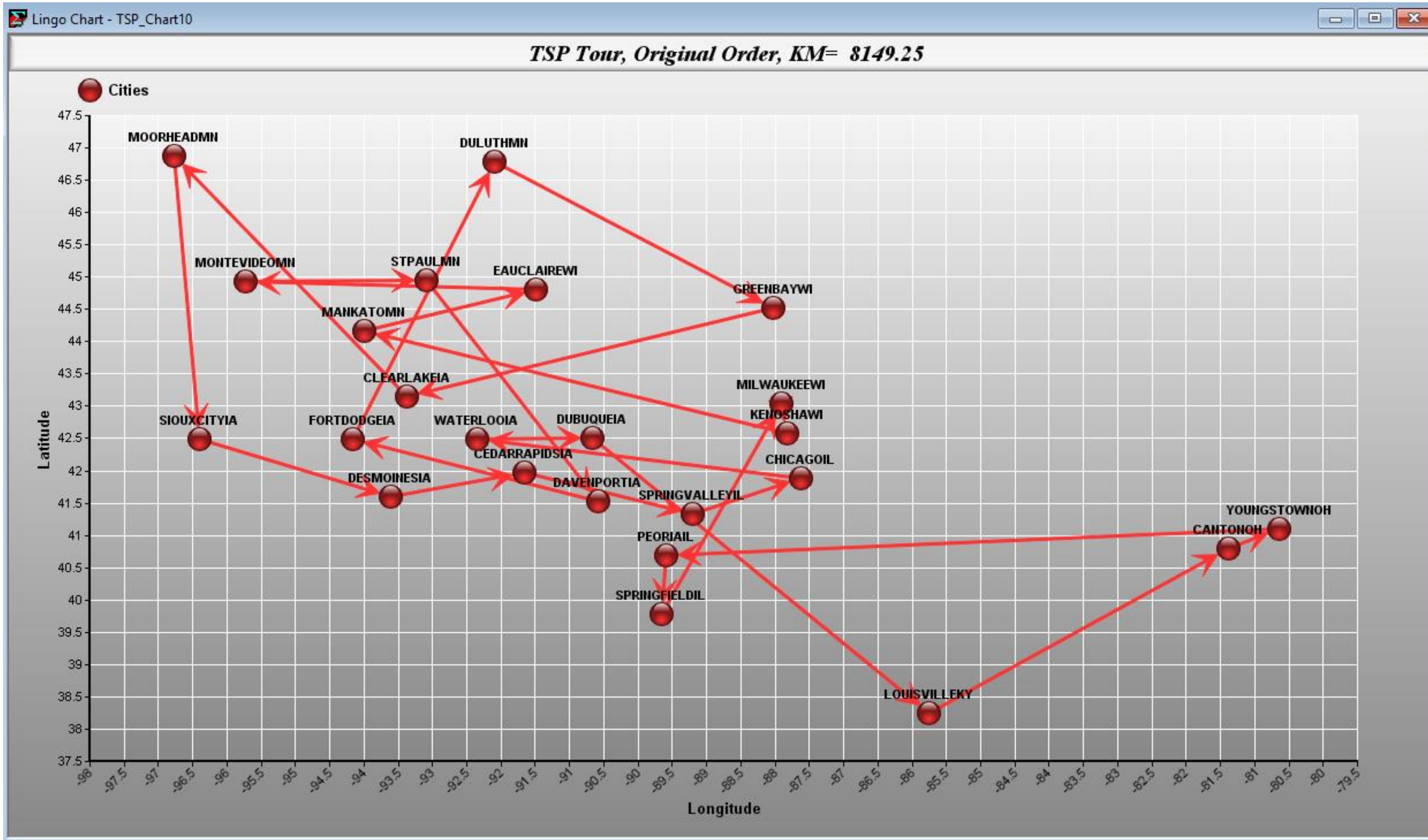
Unofficially, it was called “The Tour from Hell” because of the travel.

Not surprisingly, on February 2, 1959, Buddy Holly, Ritchie Valens, and J.P. "The Big Bopper" Richardson, rather than take a bus, decided to charter a plane from Clear Lake, IA to their next stop in Moorehead, MN.

Buddy, Ritchie, and The Big Bopper died in the plane crash.

Don McClean called it “The Day the Music Died” in his ballad, “American Pie”.

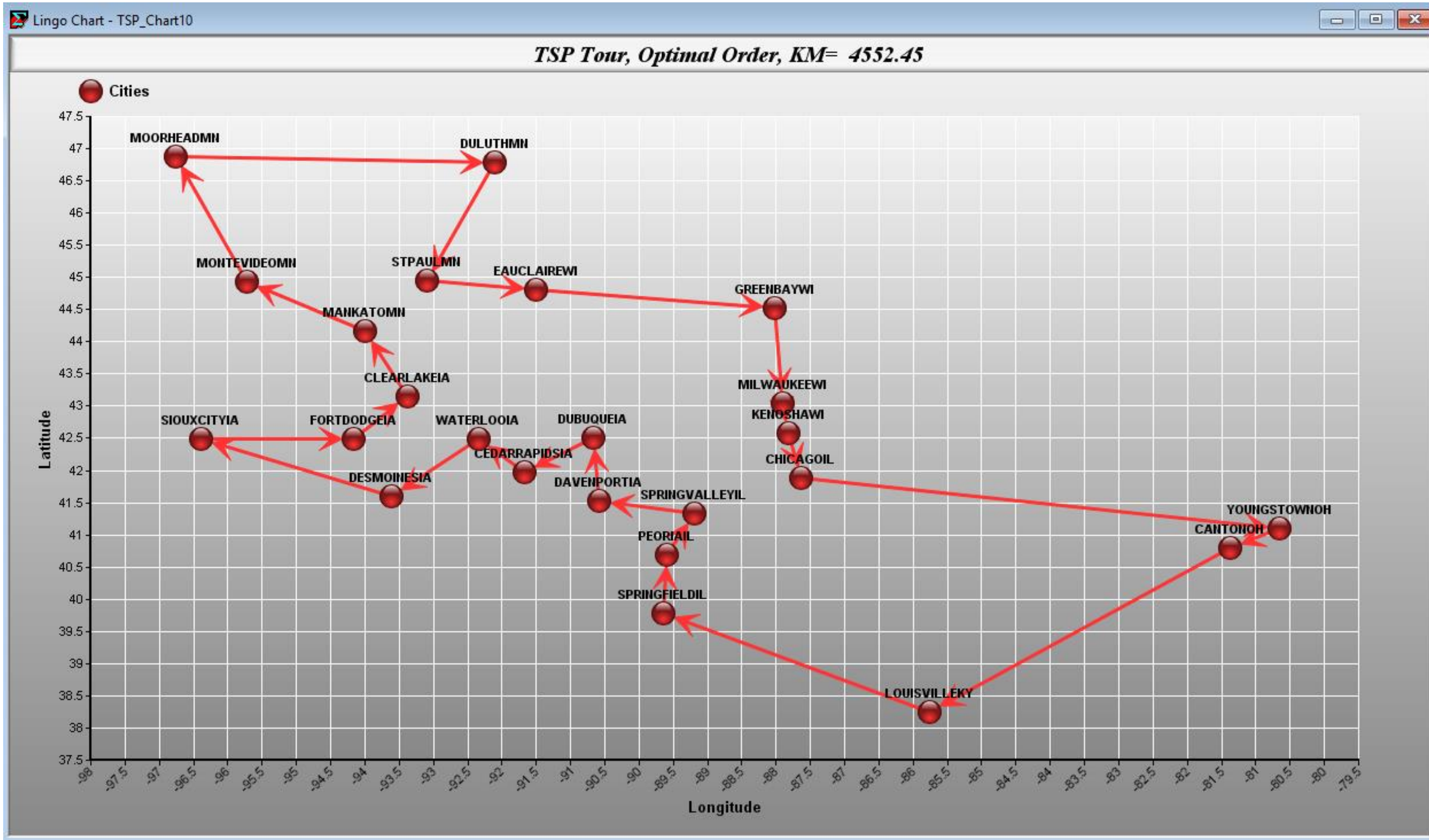
The Actual Tour Covered 8149 km (5053 miles)



Tour: Milwaukee -> Kenosha -> Mankato . .

LINDO SYSTEMS INC.

The Shortest TSP To 4552 km (2823 miles)



LINDO SYSTEMS INC.

Here is Part of the LINGO Code…

```
 ! Traveling Salesman Problem.   (TSP_Chart11.lng)
 Find the shortest tour that visits each city exactly once.
 ! The Miller, Tucker, Zemlin, 1960, J. ACM, single commodity formulation;
SETS:
  CITY : LATI, LNGT, LVL;
  CXC( CITY, CITY): DIST, Z;
  CXCSUB( CXC): DCITY, ACITY, ARROHD;
ENDSETS
DATA:
  ! Data can be pulled from a spreadsheet
   by using CITY, LATI, LNGT = @OLE(), with
   correspondingly named ranges in the
   open spreadsheet;
       CITY,           LATI,     LNGT=
!  Buddy Holly's "Tour from Hell" or "Winter Dance Party" (along with Ritchie Valens,
 J.P. "The Big Bopper" Richardson, Waylon Jennings,  Dion DiMucci (of the Belmonts),
 Tommy Allsup, and Carl Bunch);
!BHolly;     MilwaukeeWI    43.0389  -87.9065   ! 1959 Jan 23;
!BHolly;     KenoshaWI      42.5847  -87.8212   ! 1959 Jan 24;
!BHolly;     MankatoMN      44.1636  -93.9994   ! 1959 Jan 25;
!BHolly;     EauClaireWI    44.8113  -91.4985   ! 1959 Jan 26;
!BHolly;     MontevideoMN   44.9410  -95.7236   ! 1959 Jan 27;
!BHolly;     StPaulMN       44.9537  -93.09     ! 1959 Jan 28;
!BHolly;     DavenportIA    41.5236  -90.5776   ! 1959 Jan 29;
!BHolly;     FortDodgeIA    42.4975  -94.1680   ! 1959 Jan 30;
!BHolly;     DuluthMN       46.7867  -92.1005   ! 1959 Jan 31;
!BHolly;     GreenBayWI     44.5192  -88.0198   ! 1959 Feb 1;
!BHolly;     ClearLakeIA    43.1436  -93.3788   ! 1959 Feb 2 The day the music died;
  . . .
```

Generating a Network Chart is Relatively Easy

```
  .  .  .
@SOLVE( TSPROB);   ! Get optimal solution to Traveling Salesperson Prob;
! Z(i,j) = 1 if the tour goes from i to j;
! Construct the subset, CXCSUB(i,j), of arcs selected;
 @FOR( CXC( i,j) | Z(i,j) #GT# 0.5:
   @INSERT( CXCSUB, i, j);
    DCITY(i,j) = i;  ! Departure city;
    ACITY(i,j) = j;  ! Arrival city;
    ARROHD(i,j) = 0; ! Arrowheads(0=no, 1=yes) on this arc;
     );

@CHARTNETNODE(
  ' TSP Tour, Optimal Order, KM= ' + @FORMAT(OBJV,'8.2f') ! Chart Title;
    , 'Longitude', 'Latitude' ! Labels for horizontal and vertical axis;
    , 'Cities'                ! Legend for arc set 1;
    , LNGT, LATI              ! Coordinates of the nodes;
    , DCITY, ACITY);          ! Node pairs of arcs actually used;
```

A collection of arcs connecting nodes of network can be graphed with the **`@chartnetnode( )`** routine.

Time-Space Diagrams & Jet-Taxi Routing

For some routing problems, timing or “time windows” are important.

The Taxi-Routing or Full Truck Load (FTL) routing problem is essentially the following:

We are given a set of trips (typically pick-up and drop-off) that must be made, including the time that each pickup must be made.

What is the best way of routing vehicles to cover the trips?

A “space-time” diagram is a useful way of understanding both the problem (time is an important element) and the solution.

Jet Taxi Routing: Space-Time Charts & Time Functions in LINGO

```
! The (Jet) Taxi Routing Problem.                          (ARouteSC23.lng)
 Given a set of desired flights or trips to be covered,
 figure out how to route planes/vehicles to cover these flights.
 Repositioning/deadheading flights are allowed at a cost.
 This model illustrate several features:
  1) Calendar routines for coordinating the times of
      flights involving different locations and time zones,
  2) The @INSERT( ) function for creating a derived set
      of repositioning/deadhead legs according to fairly arbitrary rules.
      We do not know in advance which deadhead legs might be needed.
  3) Charting/Graphing routines for displaying a network;
! Define the data structures;
SETS:
 CITY: INITA, GMTOFF, LATI, LNGT;
 LEG ;
 CXC( CITY, CITY): TRVTIM;
 ! Loaded legs;
 LODPAIR( LEG, CITY, CITY): Year, Month, Day, Hour, Minute,
      DLTIME, Y, PLFLAG,
      DCITY, ACITY, ALTIME;
 RLEG;
! Reposition city pairs;
 RPAIR( RLEG, CITY, CITY): DRTIME, U;
 DOW /SUN..SAT/;  ! Days of week;
 CARCS: ORG, DST; ! List to be created of OD pairs;
! Set of repositioning legs actually used;
 RPAIRU(RLEG, CITY, CITY): DUCITY, AUCITY, DUTIME, AUTIME ;
ENDSETS
```

Jet Taxi Routing Problem

```
! Scalar data;
!Number loaded legs available to be flown, Relative value of covering a loaded flight,
 Relative cost of a repositioning flight, Relative cost of an aircraft,
 Limit on number of aircraft used;
  NLLG, VL, RP, RA, LA =
    !@FILE('C:¥temp¥aroutingIn.txt');
    9      ! Number loaded legs available to be flown;
    1      ! Relative value of covering a loaded flight;
    0.45   ! Relative cost of a repositioning flight;
    0.7    ! Relative cost of an aircraft;
    1 ;    ! Limit on total aircraft used;

  RLEG = 1..NRPLG; ! Possible number of repositioning legs;
  LEG = 1..NLLG; ! Get data on each loaded candidate OD pair;

! Vector data;
!  The Cities, GMT offset, latitude, longitude, limit on initial aircraft;
      City,      GMTOFF,  LATI,    LNGT,   INITA=
 !  @FILE('C:¥temp¥aroutingIn.txt');
! 1; Chicago          -6   41.8500  -87.6500   0! Chicago is 6 hours behind Greenwich Mean Time;
! 2; Denver           -7   39.7392 -104.9903   1! Denver is 7 hours ...;
! 3; Tucson           -7   32.2217 -110.9258   1
! 4; Salt_Lake_City   -7   40.7500 -111.8833   0
! 5; Phoenix          -7   33.4833 -112.0667   0
! 6; Las_Vegas        -8   36.1667 -115.2000   0
! 7; Los_Angeles      -8   34.0522 -118.2428   1
;
```

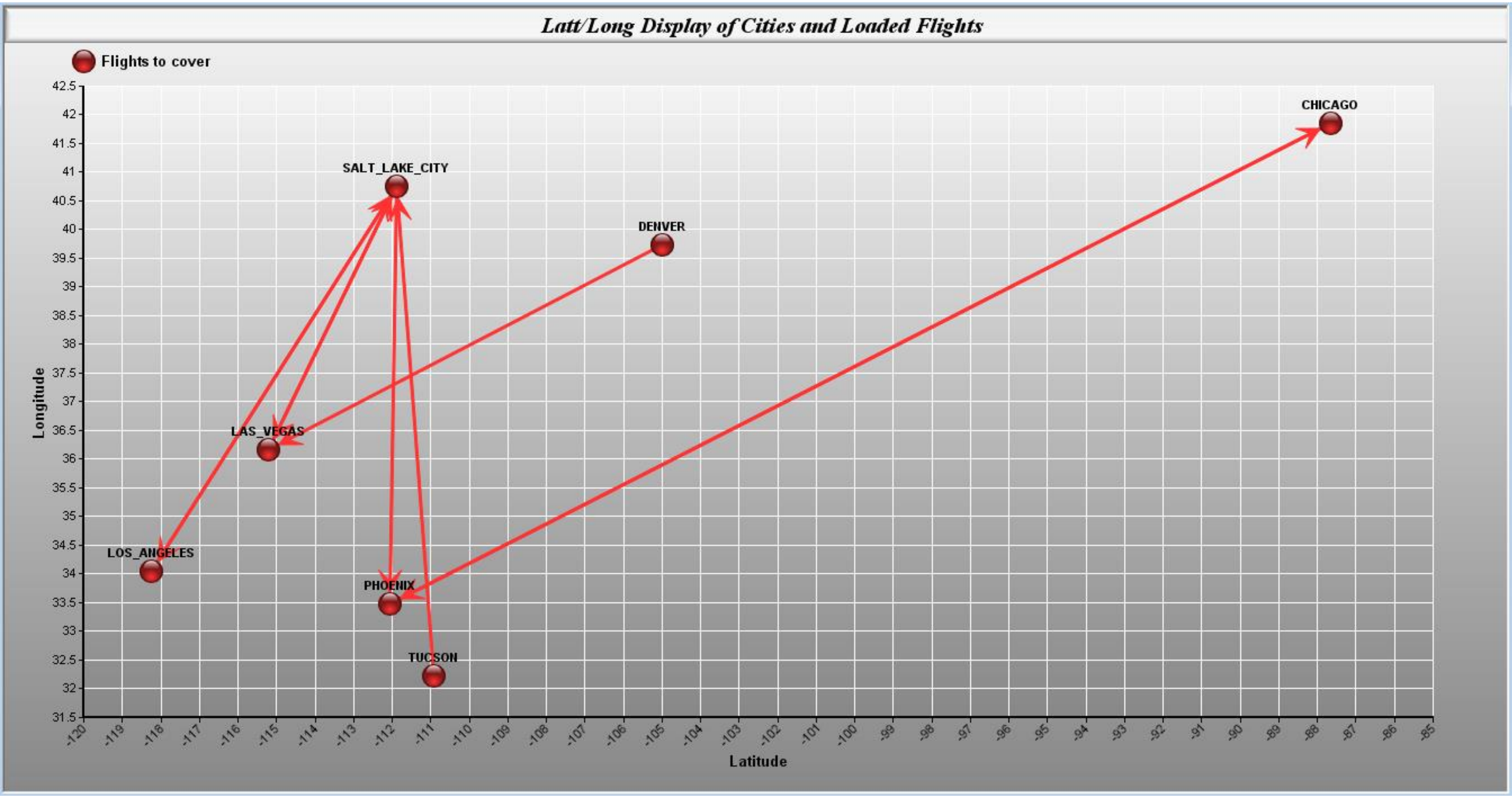
Jet Taxi Routing Problem

```
!  The city pair trips we want to cover/service;
  LODPAIR, Year, Month, Day, Hour, Minute =
!   @FILE('C:¥temp¥aroutingIn.txt');
!        Origin          Destination           Local Departure time ;
!LEG      City                   City         Year    Month   Day Hour Minute ;
  1      Los_Angeles       Salt_Lake_City  2017      4      24   10      0
  2      Salt_Lake_City    Phoenix         2017      4      25   14     20
  3      Salt_Lake_City    Los_Angeles     2017      4      27   16      0
  4      Phoenix           Chicago         2017      4      26   11     20
  5      Salt_Lake_City    Las_Vegas       2017      4      28   16      0
  6      Las_Vegas         Salt_Lake_City  2017      4      29   12      0
  7      Tucson            Salt_Lake_City  2017      4      25   15      0
  8      Denver            Las_Vegas       2017      4      26    8     30
  9      Chicago           Phoenix         2017      4      27   10     30
;
!  Get travel time matrix in minutes;
  TRVTIM =
!   @FILE('C:¥temp¥aroutingIn.txt');
! Presented by row, i.e., a list in which 'To' index moves faster than 'From';
!  Chi    Den    Tuc    SLC    Phn    LVg    LAX ;
    0    150    195    190    205    215    240 ! Chicago;
  150      0    115     85    120    115    155 ! Denver;
  195    115      0    120     60     95    120 ! Tucson;
  190     85    120      0    100     85    110 ! Salt_Lake_City;
  205    120     60    100      0     85    120 ! Phoenix;
  215    115     95     85     85      0    120 ! Las_Vegas;
  240    155    120    110    120    120      0 ! Los_Angeles;
;
ENDDATA
```

LINDO SYSTEMS INC.

Jet Taxi Routing Problem, Charting/Graphs

A standard network chart does not capture the timing aspects…



LINDO SYSTEMS INC.

Charts/Graphs: How to Find Types Available

To insert in a space-time chart routine, click on:

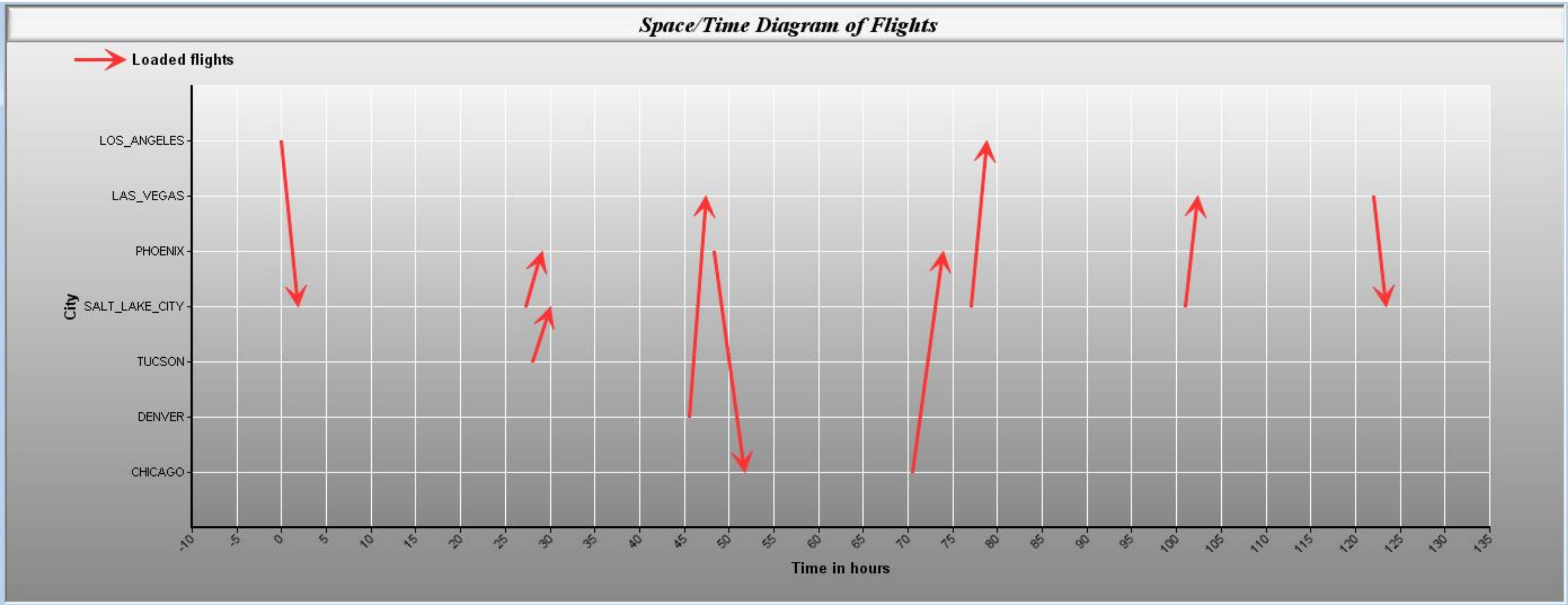
*Edit -> Paste function -> Charting -> @CHART****

Paste Function > Charting >

@CHARTBAR('title', 'x-axis', 'y-axis', 'legend1', x1[, ..., 'legendn', xn]);
@CHARTBUBBLE('title', 'x-axis', 'y-axis', 'leg1', x1, y1, size1[, ..., 'legendn', xn, yn, sizen]);
@CHARTCONTOUR('title', 'x-axis', 'y-axis', 'legend', x, y, z);
@CHARTPCONTOUR('title', 'x-axis', 'y-axis', 'z-axis', proc, x, xl, xu, y, yl, yu, 'legend', z);
@CHARTCURVE('title', 'x-axis', 'y-axis', 'legend1', x1, y1[, ..., 'legendn', xn, yn]);
@CHARTPCURVE('title', 'x-axis', 'y-axis', proc, x, xl, xu, 'legend1', y1[, ..., 'legendn', yn]);
@CHARTHISTO('title', 'x-axis', 'y-axis', 'legend', bins, x);
@CHARTLINE('title', 'x-axis', 'y-axis', 'legend1', x1[, ..., 'legendn', xn]);
@CHARTNETARC('title', 'x-axis', 'y-axis', 'legend1', x1, y1, x2, y2[, ..., 'legendn', x1n, y1n, x2n, y2n]);
@CHARTNETNODE('title', 'x-axis', 'y-axis', 'legend1', x1, y1, i1, j1[, ..., 'legendn', xn, yn, in, jn]);
@CHARTPIE('title', 'legend', x);
@CHARTRADAR('title', 'legend1', x1[, ..., 'legendn', xn]);
@CHARTSCATTER('title', 'x-axis', 'y-axis', 'legend1', x1, y1[, ..., 'legendn', xn, yn]);
@CHARTSPACETIME('title', 'x-axis', 'y-axis', 'legend1', x1, y1, x2, y2[, ..., 'legendn', x1n, y1n, x2n, y2n]);
@CHARTSURFACE('title', 'x-axis', 'y-axis', 'z-axis', 'legend', x, y, z);
@CHARTPSURFACE('title', 'x-axis', 'y-axis', 'z-axis', proc, x, xl, xu, y, yl, yu, 'legend', z);
@CHARTTORNADO('title', 'x-axis', 'y-axis', base, 'legendh', high, 'legendl', low);

Jet Taxi Routing Problem, Space-Time Diagrams

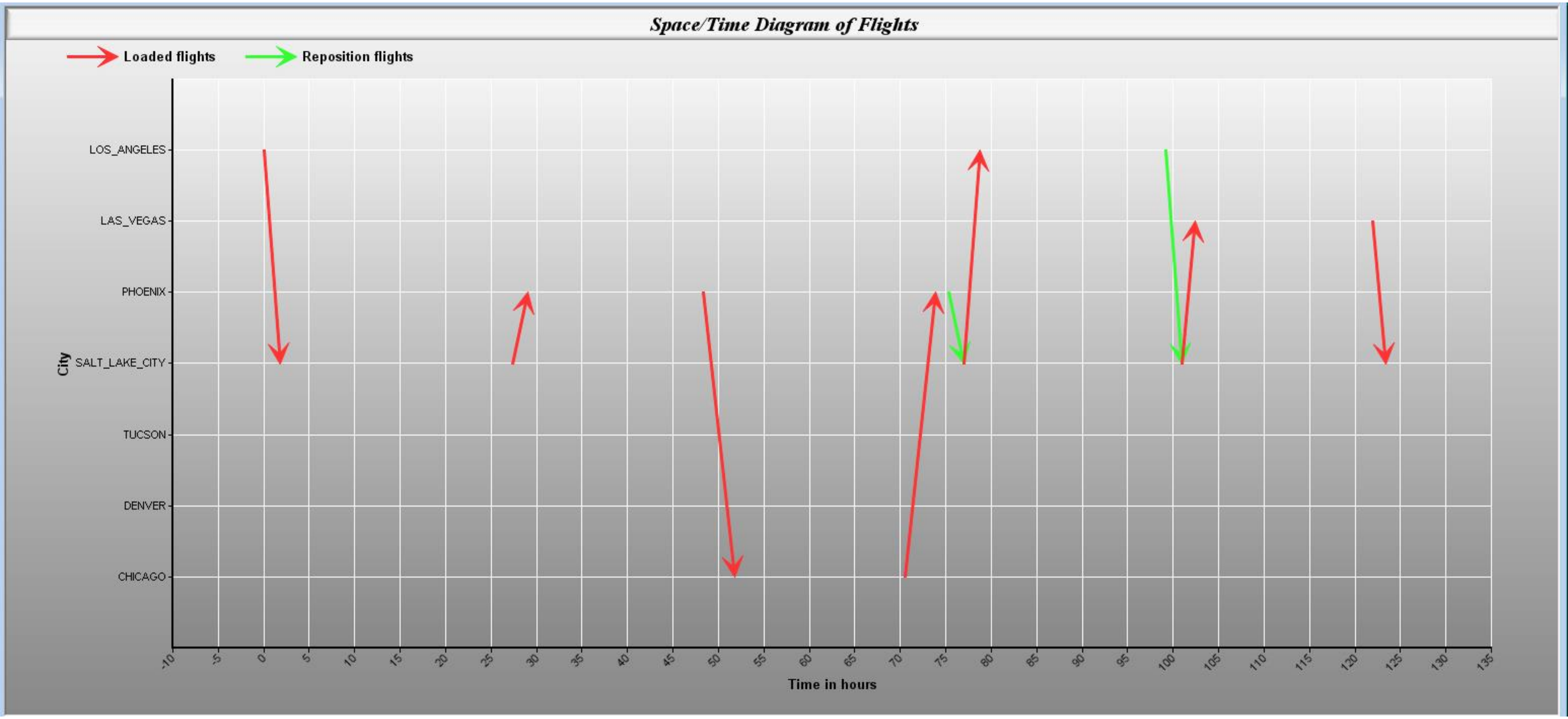
Here is a Space-Time diagram of the flights to be covered.
Time is plotted horizontally, - Cities vertically.



How many flights can you cover with one aircraft?
How many aircraft would you need to cover all these flights?

Jet Taxi Routing Problem, Space-Time Diagrams

Deadheading/Repositioning Flights Added, Using 1 Aircraft.
This is the best you can do with one vehicle for the given data.



We cover 7 of 9 possible flights.
Notice the 2 repositioning flights.

LINDO SYSTEMS INC.

Jet Taxi Routing Problem, Solution with 1 Aircraft

Value/trip covered= 1
Relative cost/repositioning= 0.45
Relative cost/aircraft= 0.7
Number aircraft allowed= 1

Number requested trips covered= 7 (of 9)
Number aircraft used= 1
Net profit contribution= 5.4

City	Lat.	Long.	GMT offset	Init. veh.:
CHICAGO	41.850	-87.650	-6	0
DENVER	39.739	-104.990	-7	0
TUCSON	32.222	-110.926	-7	0
SALT_LAKE_CITY	40.750	-111.883	-7	0
PHOENIX	33.483	-112.067	-7	0
LAS_VEGAS	36.167	-115.200	-8	0
LOS_ANGELES	34.052	-118.243	-8	1

Loaded flights selected: Depart at(local time)

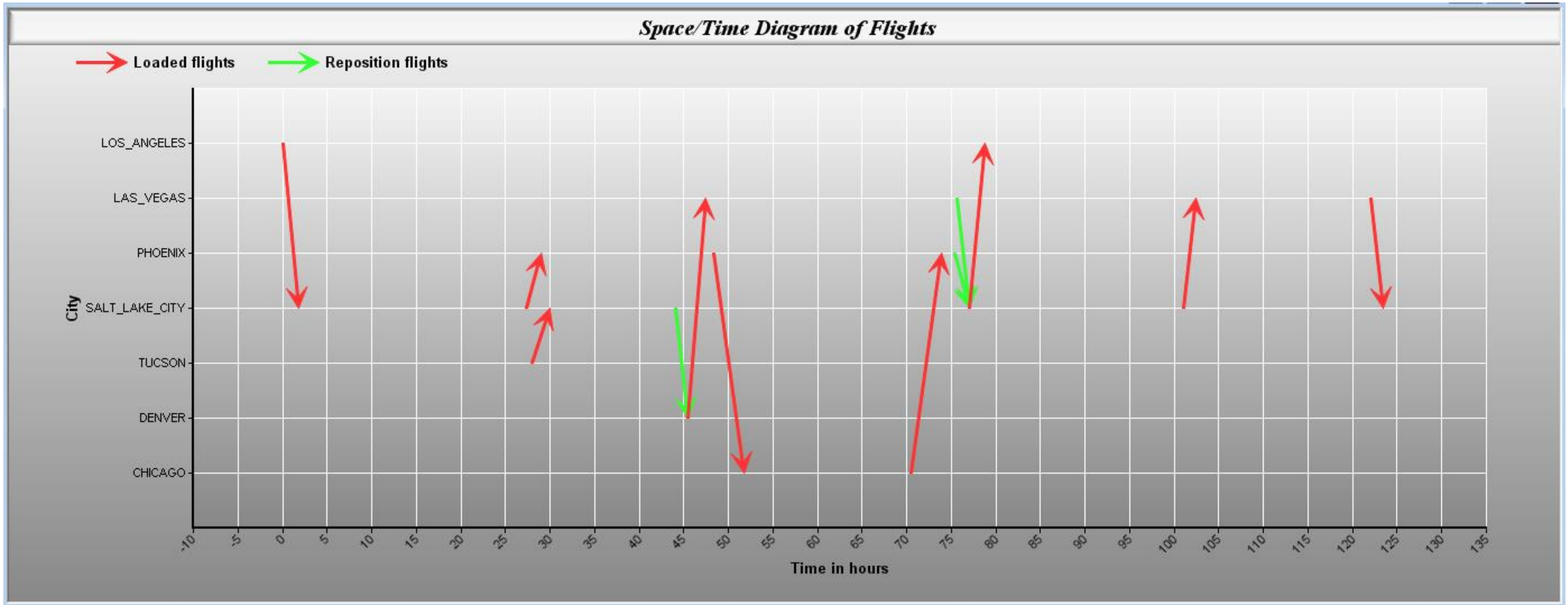
Origin	Destination	yyyy	mm	dd	hr	mm	dwk
LOS_ANGELES	SALT_LAKE_CITY	2017	4	24	10	0	MON
SALT_LAKE_CITY	PHOENIX	2017	4	25	14	20	TUE
PHOENIX	CHICAGO	2017	4	26	11	20	WED
CHICAGO	PHOENIX	2017	4	27	10	30	THU
SALT_LAKE_CITY	LOS_ANGELES	2017	4	27	16	0	THU
SALT_LAKE_CITY	LAS_VEGAS	2017	4	28	16	0	FRI
LAS_VEGAS	SALT_LAKE_CITY	2017	4	29	12	0	SAT

Repositioning Flights:

Origin	Destination	yyyy	mm	dd	hh	mm	dwk
PHOENIX	SALT_LAKE_CITY	2017	4	27	14	20	THU
LOS_ANGELES	SALT_LAKE_CITY	2017	4	28	13	10	FRI

Jet Taxi Routing Problem

Deadheading/Repositioning Flights Added, Using 2 Aircraft



We cover all 9 desired flights.
We need 3 re-positioning flights.

LINDO SYSTEMS INC.

Jet Taxi Routing Problem, Solution with 2 Aircraft

```
Value/trip covered= 1
Relative cost/repositioning= 0.45
Relative cost/aircraft= 0.7
Number aircraft allowed= 2

Number requested trips covered= 9 (of 9)
Number aircraft used= 2
Net profit contribution= 6.25
           City          Lat.       Long.  GMT offset Init. veh.:
          CHICAGO      41.850    -87.650      -6        0
           DENVER      39.739   -104.990      -7        0
           TUCSON      32.222   -110.926      -7        1
   SALT_LAKE_CITY      40.750   -111.883      -7        0
          PHOENIX      33.483   -112.067      -7        0
        LAS_VEGAS      36.167   -115.200      -8        0
      LOS_ANGELES      34.052   -118.243      -8        1

Loaded flights selected:                 Depart at(local time)
   Origin              Destination       yyyy mm  dd hr mm  dwk
      LOS_ANGELES       SALT_LAKE_CITY   2017  4  24 10  0  MON
   SALT_LAKE_CITY              PHOENIX   2017  4  25 14 20  TUE
           TUCSON       SALT_LAKE_CITY   2017  4  25 15  0  TUE
           DENVER            LAS_VEGAS   2017  4  26  8 30  WED
          PHOENIX              CHICAGO   2017  4  26 11 20  WED
          CHICAGO              PHOENIX   2017  4  27 10 30  THU
   SALT_LAKE_CITY          LOS_ANGELES   2017  4  27 16  0  THU
   SALT_LAKE_CITY            LAS_VEGAS   2017  4  28 16  0  FRI
        LAS_VEGAS       SALT_LAKE_CITY   2017  4  29 12  0  SAT

Repositioning Flights:
   Origin              Destination       yyyy mm  dd hh  mm dwk
   SALT_LAKE_CITY               DENVER   2017  4  26  7   5 WED
        LAS_VEGAS       SALT_LAKE_CITY   2017  4  27 13  35 THU
          PHOENIX       SALT_LAKE_CITY   2017  4  27 14  20 THU
```

LINDO SYSTEMS INC.

Jet Taxi Routing Problem, the Optimization Model

```
SUBMODEL ROUTEM:
 ! Variables:
      Y(n,d,a) = 1 if we do flight leg n,
                     from city d to city a,
      U(n,d,a) = 1 if we do the nth possible deadheading flight,
                     from city d to city a;

 ! Maximize number of requested flights flown
  - cost of repositioning flights
  - cost of aircraft;
  MAX = VL*@SUM( LODPAIR( n, d, a): Y(n,d,a)) ! Loaded flights;
      - RP*@SUM( RPAIR( n, d, a): U(n,d,a)) ! Repositions;
      - RA*@SUM( CITY(i): INITA(i)); ! Initial AC at city i;

 ! You either fly it or you do not;
  @FOR( LODPAIR( n, d, a): @BIN(Y(n,d,a)));
  @FOR( RPAIR( n, d, a): @BIN(U(n,d,a)));

 ! For every departing loaded flight from d to a at time DLTIME,
  the number of earlier arrivals - earlier departures must be >= Y(n,d,a);
 @FOR( LODPAIR( n, d, a):
  [LFLO] INITA(d)                ! Note, scalar time is in seconds, not minutes;
  + @SUM( LODPAIR( n1, d1, d) | DLTIME(n1,d1,d) + TRVTIM(d1,d)*60 #LE# DLTIME(n,d,a):
       Y(n1,d1,d))  ! loaded flights into d;
  + @SUM( RPAIR( n1, d1, d) | DRTIME(n1,d1,d) + TRVTIM(D1,d)*60 #LE# DLTIME(n,d,a):
       U(n1,d1,d))  ! Dead-head (unloaded) flights into d;
  - @SUM(LODPAIR(n1,d,a1) | DLTIME(n1,d,a1) #LT# DLTIME(n,d,a):
       Y(n1,d,a1))  ! Loaded flights out of d;
  - @SUM( RPAIR(n1,d,a1) | DRTIME(n1,d,a1) #LE# DLTIME(n,d,a):
       U(n1,d,a1))  ! Dead head flights out;
   >= Y(n,d,a);
     );
```

Jet Taxi Routing Problem

```
CALC:
  @SET( 'TERSEO', 2);  ! Set output to terse;
  ! Convert Year, Month, Day, Hour, Minute(n,d,a) to a scalar DLTIME(n,d,a)
    so we can do simple comparisons and arithmetic;
  ! Compute departure time in scalar GMT time for each loaded flight;
  @FOR( LODPAIR( n, d, a):
    DLTIME(n,d,a) = @YMD2STM( Year(n,d,a), Month(n,d,a), Day(n,d,a) , Hour(n,d,a), Minute(n,d,a), 0)
                    - 3600*GMTOFF(d); ! Take into account local time, convert hours to seconds;
       );

 ! Construct the set of candidate repositioning legs. For each departing
  flight from city j, we add a candidate repositioning leg from every
  other city i, j #NE# i, at TRVTIM(i,j) minutes earlier ;
  k = 0;
  @FOR( LODPAIR( n, j, a):
    @FOR( CITY(i) | i #NE# j:
      k = k+1;
  ! Insert into the RPAIR set, the triple (k, i, j);
      @INSERT( RPAIR, k, i, j);
  ! Now that the element has been added to the set, we can set its attributes;
      ! Note, travel times are in minutes, scalar TRVTIM(i,j) in seconds;
      DRTIME(k,i,j) = DLTIME(n,j,a) - TRVTIM(i,j)*60;
        );
      );
```

Jet Taxi Routing Problem

```
@WRITE(@NEWLINE(1),' Repositioning Flights:',@NEWLINE(1));
@WRITE('    Origin                    Destination          yyyy  mm   dd hh   mm
dwk',@NEWLINE(1));

@FOR( RPAIR( n, d, a) | U(n,d,a) #GT# 0.5:
  ! Convert DRTIME(n,d,a) back to year month, day, hour minute;
    CTIME = DRTIME(n,d,a)+ 3600*GMTOFF(d); ! Take into account local time,
convert hrs to secs;
    IYR, IMON, IDAY, IWKD, IHR, IMIN, ISEC = @STM2YMDHMS( ctime);

    @WRITE( @FORMAT(CITY(d),'18s'),'   ',@FORMAT(CITY(a),'18s'),'   ',IYR,'  ',
    IMON,'  ',@FORMAT(IDAY,'2.0F'),' ',@FORMAT(IHR,'2.0F'),'
',@FORMAT(IMIN,'2.0F'),' ',DOW(IWKD),@NEWLINE(1));
    );

! Build vectors to prepare to draw various networks;
   @FOR( LODPAIR(n,d,a):
     DCITY(n,d,a) = d;  ! Departure city of leg;
     ACITY(n,d,a) = a;  ! Arrival city of leg;
! Time that a flight arrives at destination city;
     ALTIME(n,d,a) = DLTIME(n,d,a) + TRVTIM(d,a)*60;
       );

! If we want to plot;
  PLOTIT;
! Write Flat file of input and output;
!  WriteFlatFile;
 ENDCALC
```

LINDO SYSTEMS INC.

In Parametric Analysis we analyze, frequently graphically, how changing one parameter causes a change in some other measure. Typical trade offs are:

Increasing desired expected return on investment
increases risk;

Increasing advertising budget
increases number of "eyes" seeing our products.

Parametric Analysis: Markowitz Portfolio

Efficient Frontier Portfolio Calculation (See PortEfFront11.lng)

The possible investments:

CD___= risk-free rate,
VG040= SP500 stock index,
VG058= Insured long term tax exempt,
VG072= Pacific stock index
VG079= European Stock index,
VG102= Tax managed cap appreciation,
VG533= Emerging markets.

After tax Target Return	Risk(1 sd) 1-Yr	Portfolio composition CD___	VG040	VG102	VG058	VG079	VG072	VG533
0.04000	0.0000	1.0000						
0.04500	0.0106	0.6483	0.0085	0.0265	0.2496	0.0413	0.0257	
0.05000	0.0212	0.2967	0.0170	0.0530	0.4993	0.0827	0.0513	
0.05500	0.0321		0.1052		0.6645	0.1316	0.0987	
0.06000	0.0541		0.0282		0.5349	0.0926	0.1999	0.1443
0.06500	0.0806				0.4046		0.2863	0.3091
0.07000	0.1087				0.2155		0.3535	0.4310
0.07500	0.1376				0.0264		0.4207	0.5528
0.08000	0.1733							1.0000

Input Data Used:

Expected ret/yr:	0.0400	0.0600	0.0600	0.0500	0.0650	0.0700	0.0800
Stdev in ret/yr:	0.0000	0.0811	0.0911	0.0370	0.1010	0.1252	0.1733

Parametric Analysis


```
! Graph it as is done by Finance folks;
  @CHARTCURVE( 'Return vs Risk', 'Return', 'Risk', 'Standard Deviation', VOUT, VINP);
```

Tornado Diagrams for Analyzing Causes of Uncertainty

Essential Idea:

There are 2 or more parameters that make our life unpredictable. E.g.:

Market price for our product, e.g., crude oil,

Demand for our product,

Productivity of our labor,

Availability of various key raw materials.

For each source of uncertainty we specify there possible possibilities:

1) Lowest possible, 2) Most likely, 3) Highest possible.

For each source of uncertainty:

Analyze the two extreme cases,

Rank sources by impact on profit.

Tornado Diagram: A Specific Case

```
! Tornado parametric analysis of AstroCosmo model. (AstroCosTrndo.lng)

 We are about to produce two products, Astros and Cosmos.
There are seven parameters of which we are unsure:
 1:2) The profit contribution of each (mainly selling price),
 3:4) Production line capacities for each of the two lines,
   5) Total amount of labor available per day.
 6:7) Labor usage rate/unit for each product,
For each parameter we estimate a
lowest possible value, PLO, a most likely median value, PMED,
and a highest possible value, PHI. We are interested how the
uncertainty in parameter value affects total profit.
The analysis identifies for each parameter, the "bottom line"
uncertainty resulting from the input uncertainty of each parameter.

The Tornado diagram (it looks like a tornado/funnel cloud) gives a
graphical display,
         most uncertainty causing parameter at the top,
         least uncertainty causing parameter at the bottom;
```

Tornado Diagram Analysis

Recall: Parametric/Uncertain/Scenario Case:

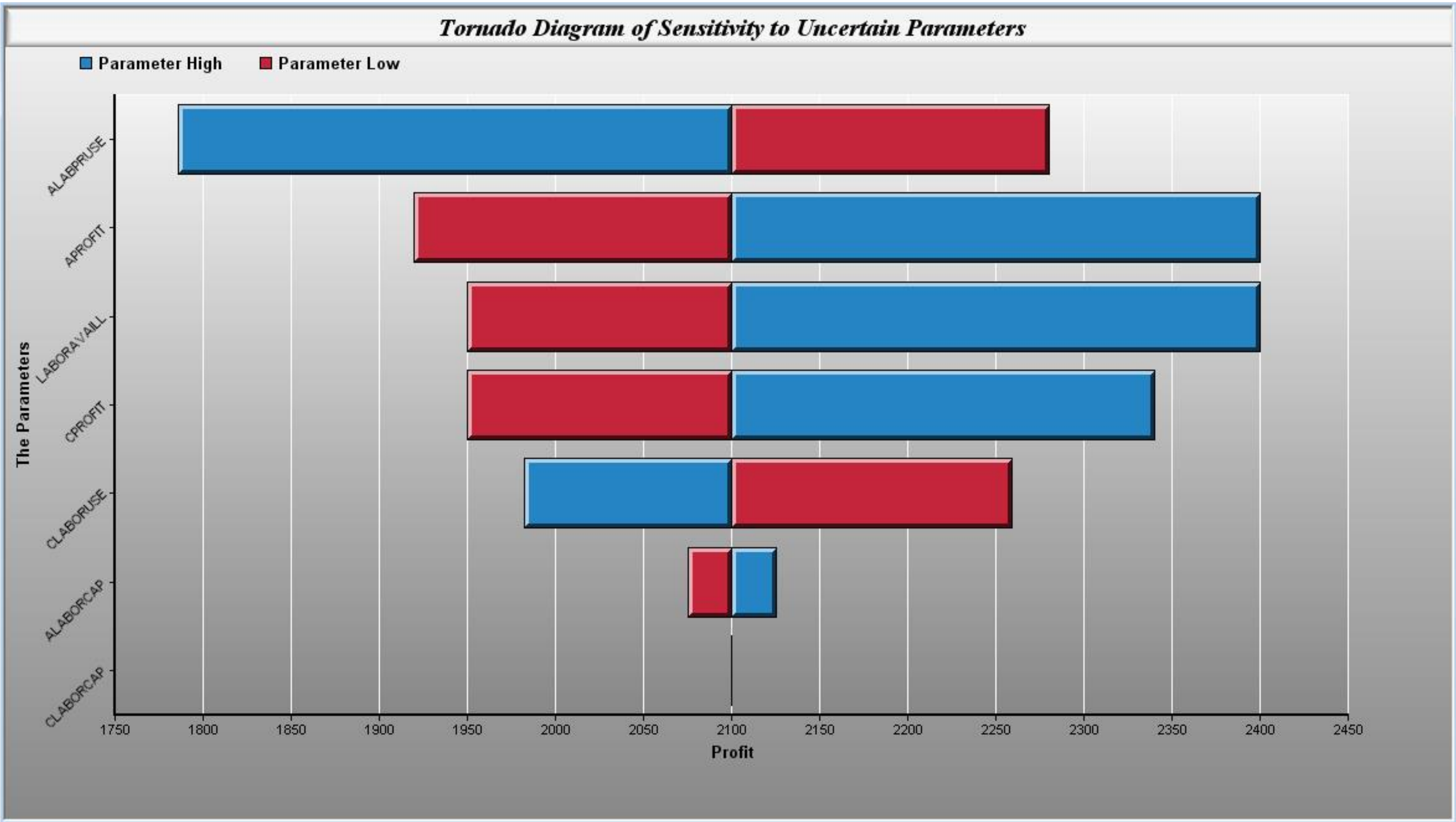
```
 MAX = PAM(1)*ASTRO + PAM(2)*COSMO;
          ASTRO <= PAM(3);  ! Astro demand;
          COSMO <= PAM(4);  ! Cosmo demand;
        PAM(6)*ASTRO + PAM(7)*COSMO <= PAM(5);
```

```
DATA:
 ! Names of random parameters;
  PSET =
    APROFIT CPROFIT ALABORCAP CLABORCAP LABORAVAIL ALABPRUSE CLABORUSE ;
 ! The median or base case values for the parameters;
  PMED =
       20      30       60         50          120           1          2 ;
 ! Plausible low values for the parameters;
  PLO =
       17      25       55         45          110         0.8        1.7 ;
 ! Plausible high values for the parameters;
  PHI =
       25      38       65         60          140         1.4        2.3 ;
! For this parameter set we will see that LABORAVAIL has
 the greatest effect on bottom line uncertainty.
 CLABORCAP has the least effect (none) on bottom line uncertainty;
ENDDATA
```

LINDO SYSTEMS INC.

Tornado Diagram Analysis

```
@CHARTTORNADO( 'Tornado Diagram of Sensitivity to Uncertain Parameters',
'Profit', 'The Parameters', BASE,
'Parameter High', RESULTHI, 'Parameter Low', RESULTLO);
```



Basic idea:

Given values of various characteristics (“explanatory variables”) of an object, predict its category, e.g.,

Is a prospective customer a good credit risk, or bad?

Is a paper banknote good or counterfeit?

Does a patient have a certain disease or not?

We want to compute the weights on the explanatory variables in a scoring formula, so that

Score(i) ≥ 0 implies a good item, < 0 implies bad.

There are various objectives one can use in selecting a scoring function. Here we use the objective of

Minimize the number of misclassifications;

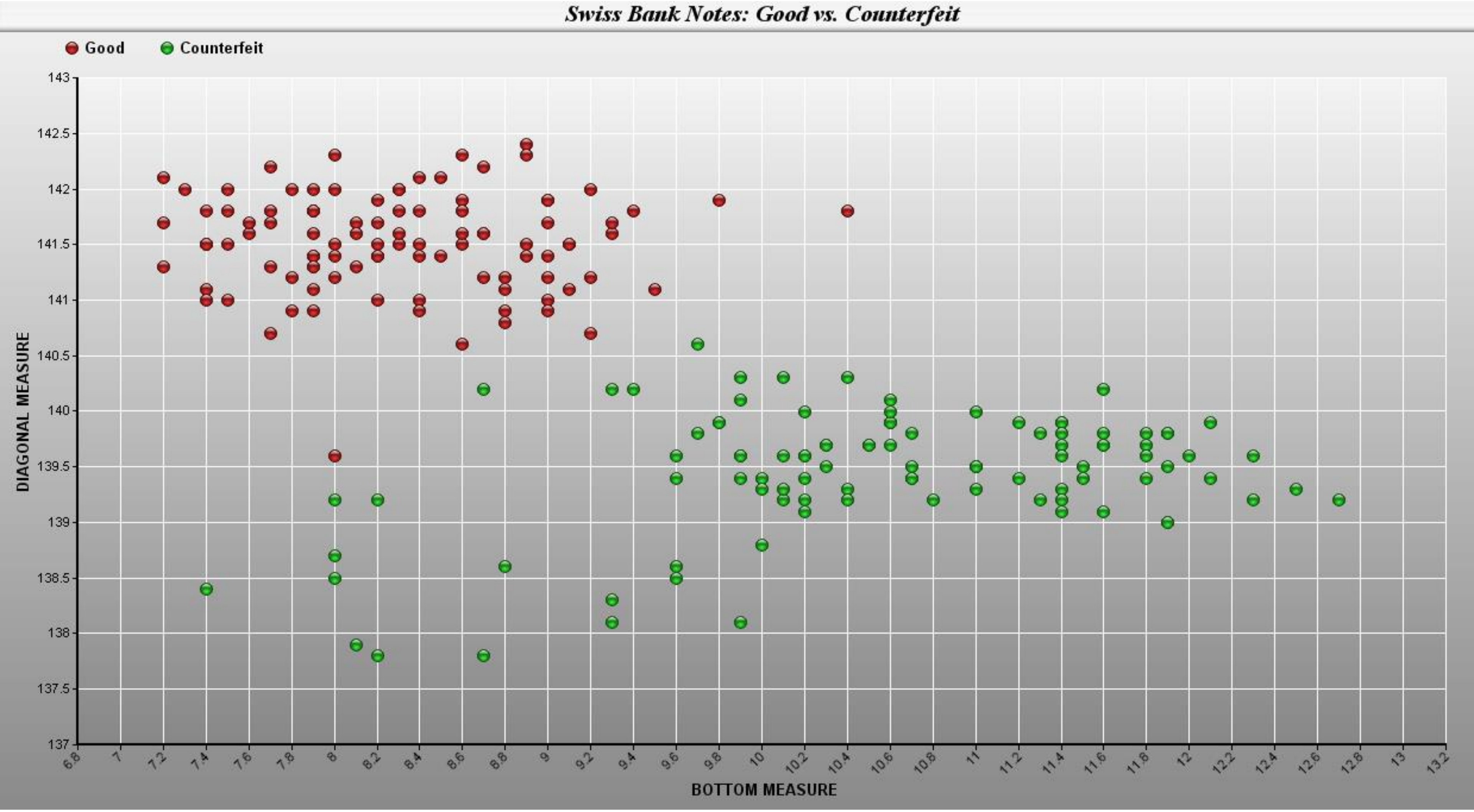
Discriminant Analysis, Example

We have 200 observations on a mix of good and counterfeit Swiss bank notes. For each banknote, we have six measurements, and we know whether good or counterfeit.

	Length	Left	Right	Bottom	Top	Diagonal	Good;
BN1	214.8	131.0	131.1	9.0	9.7	141.0	1
BN2	214.6	129.7	129.7	8.1	9.5	141.7	1
BN3	214.8	129.7	129.7	8.7	9.6	142.2	1
BN4	214.8	129.7	129.6	7.5	10.4	142.0	1
BN5	215.0	129.6	129.7	10.4	7.7	141.8	1
. . .							
BN70	214.9	130.2	130.2	8.0	11.2	139.6	1
. . .							
BN103	214.9	130.3	130.1	8.7	11.7	140.2	0
. . .							
BN195	214.9	130.3	130.5	11.6	10.6	139.8	0
BN196	215.0	130.4	130.3	9.9	12.1	139.6	0
BN197	215.1	130.3	129.9	10.3	11.5	139.7	0
BN198	214.8	130.3	130.4	10.6	11.1	140.0	0
BN199	214.7	130.7	130.8	11.2	11.2	139.4	0
BN200	214.3	129.9	129.9	10.2	11.5	139.6	0

Qualitatively, "Good" seems associated with large Diagonal and small Bottom measurement.

Discriminant Analysis, Scatter Plot of Two Dimensions



With a graph, the separation of "Good" from "Counterfeit" is quite clear.

LINDO SYSTEMS INC.

Discriminant Analysis: Some of the Code

```
! Discriminant analysis by integer programming (DiscrmSwiss.lng);
SETS:
 TEST: WGT, ZUSE;
 OBS: DROP, SCORE;
 OXT(OBS, TEST): TSCR;
 OBS1(OBS): X1, Y1;
 OBS2(OBS): X2, Y2;
ENDSETS
DATA:
!  Genuine and counterfeit banknotes (100 Swiss Franks),
various measurements. Dataset courtesy of H. Riedwyl, Bern,
Switzerland;
 WGTSUSEDMX = 2;  ! Max # of weights to use;
 WGTMX = 99999;   ! Max absolute value of any weight;
 DEPVAR = 7;      ! Index of the dependent variable (Good);
 TEST=
      Length  Left    Right   Bottom  Top     Diagonal  Good;
 OBS, TSCR=
BN1      214.8   131.0   131.1   9.0     9.7     141.0     1
BN2      214.6   129.7   129.7   8.1     9.5     141.7     1
BN3      214.8   129.7   129.7   8.7     9.6     142.2     1
        .  .  .
```

Discriminant Analysis: Some of the Code

```
SUBMODEL DISCRAMP:
! Minimize number of observations dropped to get a partition;
  MIN = OBJV;
    OBJV = @SUM( OBS( I): DROP( I));
! For bad observations, if DROP(I)=0, we want a strictly negative score;
@FOR( OBS(I)| TSCR( I, DEPVAR) #EQ# 0:
   SCORE( I) <= -1 + WGTMX*DROP( I);
   SCORE( I) >=    - WGTMX*(1- DROP(I));
   SCORE( I) =
   WGT0 + @SUM( TEST( J) | J #NE# DEPVAR: WGT( J)* TSCR(I,J));
   @FREE( SCORE(I));
    );
! For good observations, if DROP(I)=0, we want a strictly positive score;
@FOR( OBS(I)| TSCR( I, DEPVAR) #EQ# 1:
   SCORE( I) >= 1 - WGTMX*DROP( I);
   SCORE( I) <= WGTMX*(1-DROP(I));
   SCORE( I) =
   WGT0 + @SUM( TEST( J) | J #NE# DEPVAR: WGT( J)* TSCR(I,J));
   @FREE( SCORE(I));
    );
 @FREE( WGT0);
 @FOR( TEST( J): @FREE( WGT( J));); ! The WFT(J) are unrestricted in sign;
 @FOR( OBS(I): @BIN( DROP(I))        ! The DROP(I) are 0 or 1;
     );
! Constraints limit number of nonzero weights;
 @FOR( TEST( K) | K #NE# DEPVAR:
    WGT( K) <= WGTMX*ZUSE( K);
   -WGT( K) <= WGTMX*ZUSE( K);
    @BIN( ZUSE( K));
     );
  @SUM( TEST( K) | K #NE# DEPVAR: ZUSE( K)) <= WGTSUSEDMX;
ENDSUBMODEL
```

LINDO SYSTEMS INC.

Discriminant Analysis: Some of the Code

```
! Create set of the GOOD ones, with 2 dimensions, D1 and D2, in X1, Y1;
  @FOR( OBS(I) | TSCR( I, DEPVAR) #EQ# 1:
    @INSERT( OBS1, I);
    X1( I) = TSCR(I,D1);
    Y1( I) = TSCR(I,D2);
      );
! Create set of the BAD ones, with 2 dimensions in X2, Y2;
  @FOR( OBS(I) | TSCR( I, DEPVAR) #EQ# 0:
    @INSERT( OBS2, I);
    X2( I) = TSCR(I,D1);
    Y2( I) = TSCR(I,D2);
      );

 @WRITE( ' Measure       WGT', @NEWLINE(1));
 @WRITE( ' CONSTANT ', @FORMAT( WGT0, '10.3f'), @NEWLINE(1));
 @FOR( TEST( J) | J #NE# DEPVAR:
   @WRITE( @FORMAT( TEST( J),'9s'), @FORMAT( WGT( J), '10.3f'), @NEWLINE(1));
     );
  @WRITE( @NEWLINE(1));
  @WRITE(' If CONSTANT + @SUM( TEST( j): WGT(j)*TSCR(i,j)) >= 0,', @NEWLINE(1));
  @WRITE('       Then predict as GOOD, else Predict as BAD.', @NEWLINE(1));
  @WRITE( @NEWLINE(1),'Number items incorrectly predicted= ', OBJV, @NEWLINE(1));

! Now do a scatter plot;
  @CHARTSCATTER(
    'Swiss Bank Notes: Good vs. Counterfeit', !Chart title;
    @FORMAT(TEST(D1),"7s")+' MEASURE', !Legend for X axis;
    @FORMAT(TEST(D2),"7s")+' MEASURE', !Legend for Y axis;
    'Good', x1, y1,            !Point set 1;
    'Counterfeit', x2, y2);  !Point set 2;
```

Discriminant Analysis: Score Function Using 2 Explanatory Variables

Measure	WGT
CONSTANT	-6347.800
LENGTH	0.000
LEFT	0.000
RIGHT	0.000
BOTTOM	-44.000
TOP	0.000
DIAGONAL	48.000

```
If CONSTANT + @SUM( TEST( j): WGT(j)*TSCR(i,j)) >= 0,
      Then predict as GOOD, else Predict as BAD.

Number items incorrectly predicted= 0

    For example for GOOD point B70:
       -6347.8 - 44*8   + 48*139.6 = +1;
    For example for BAD point B103:
       -6347.8 - 44*8.7 + 48*140.2 = -1;
```