mixing.mws

Contour Plots for Three-Ingredient Mixing Problems

Normal Plots of Residuals

Full-Spectrum Contour Plots

Carl Devore <devore@math.udel.edu>

27 May 2001

In this worksheet, I consider a class of experiments where the "independent" variables are the *relative* proportions of three ingredients in a mixture. I put "independent" in quotes because we have the constraint that the proportions sum to unity, so there are really only two independent variables. Let me define *three-ingredient space* as that portion of the plane x + y + z = 1 that lies in the first octant. This is an equilateral triangle with vertices at (1,0,0), (0,1,0), and (0,0,1). We measure some response, Y, at a variety points in the space and attempt to statistically fit a function to the data. The level curves (also known as contours) of this function will be plotted in three-ingredient space.

Several functions and ideas are explored in this worksheet:

1. Plotting in the three-ingredient space. I am not aware of other mathematical or statistical software that provides this plot.

2. Normal plots of the residuals from linear regression. I believe that the procedure that I present gives more useful information than the equivalent MINITAB command.

3. Contour plots using the full spectrum of colors. This gives a much easier-to-read correspondence between the colors and the numbers they represent than does the color scaling in Maple's countourplot command.

Other topics dealt with briefly:

4. Computing 3d plotting angles from lines of sight.

5. Manipulating the values printed on tickmarks.

6. Selecting rows and columns from the new rtable-based Matrices.

7. Rearranging expressions to avoid catastrophic cancellation.

And I present brief procedures for

8. F-test for significance of regression (also known as ANOVA: ANalysis Of VAriance).

9. Finding the data points corresponding to outlying residuals.

If you use this worksheet, please send me email <devore@math.udel.edu> to discuss it, to make suggestions, or just to say "hi". I am frequently willing to write Maple programs if the topic interests me. I can be reached via paper mail at

Carl Devore

655 E3 Lehigh Rd

Newark DE 19711

USA

or by phone at (302) 738-2606 or (302) 831-3176. But email is the best way to reach me.

Caveat: I am not a statistician. If I have made any statistical errors in the following, please inform me.

```
> restart;
```

```
> Digits:= trunc(evalhf(Digits));
```

$$Digits := 14$$

```
> UseHardwareFloats:= true:
```

Some abbreviations that are useful for all matrix work:

```
> `&T`:= LinearAlgebra:-Transpose:
```

```
> `&L`:= X -> convert(X,list):
```

```
> `&LL`:= X -> convert(X, listlist):
```

```
> `&/`:= (X,Y) -> LinearAlgebra:-LeastSquares(Y,X):
```

We want to use the perpendiculars of the triangle as the coordinate axes, each being scaled from 0 to1. Since each side of the triangle corresponds to a variable being set to 0, each axis will go from 1 at a vertex to 0 at the opposite side. It is theoretically possible to do this with 2-dimensional plotting commands, but it would make the handling of the axes and tickmarks difficult, and we would have to use a complicated projection. With a few tricks, we can get Maple's 3-dimensional plotting commands to do most of the work. Then all we have to do is plot in three-ingredient space and everything appears natural.

The main trick is to take a 3d plot of three-ingredient space and view it at an angle that causes the coordinate axes to appear as the perpendiculars of the equilateral triangle. Thus, we want the axes to appear as if they intersect at the centroid of the triangle, the point (1/3, 1/3, 1/3). Imagine viewing this from the octant where all coordinates are negative, so that the axes are between you and the triangle. We want the line from (1/3, 1/3, 1/3) to (0,0,0) to be a line of sight, so you can imagine that you are viewing from the point (-1,-1,-1), which is on that line. Maple's plots work in terms of spherical-coordinate angles rather than viewpoints or sightlines. We convert the point (-1, -1, -1) to spherical coordinates.

```
> `xyz->spherical`:=
(x,y,z) -> (sqrt(x^2+y^2+z^2), arctan(x,y), Pi/2 - arctan(z, sqrt(x^2+y^2)));
```

$$xyz\text{->}spherical := (x, y, z) \rightarrow \left(\sqrt{x^2+y^2+z^2}, \arctan(x, y), \frac{1}{2}\pi - \arctan(z, \sqrt{x^2+y^2})\right)$$

```
> rho, theta, phi:= `xyz->spherical`(-1,-1,-1);
```

$$\rho, \theta, \phi := \sqrt{3}, -\frac{3}{4}\pi, \frac{1}{2}\pi + \arctan\left(\frac{1}{2}\sqrt{2}\right)$$

Maple requires the angles in plot commands to be in degrees.

```
> theta,phi:= evalf(map(x-> x*180/Pi, [theta,phi]))[];
```

$$\theta, \phi := -135., 125.26438968276$$

Next, we need to make the axes the right length. The corners of the triangle are at position 1 on each axis. The origin appears to be at (1/3, 1/3, 1/3). It follows that to get the axes to extend to the opposite sides of the triangle, we need to extend each axis by half of its original length. So each axis goes from -1/2 to 1.

```
> axes_range:= -0.5..1:
```

Now we need to change the values printed on the tickmarks. We want to map 0..1 to -1/2..1, so the transformation is (3*t-1)/2.

First, make a list of the values that we want printed on the axes. I like .1, .2, ..., .9.

```
> ts:= [seq(.1*i, i= 1..9)]:
```

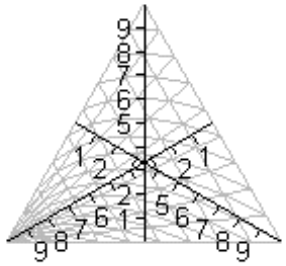
The decimal points just add unnecessary clutter to the plot. I remove them. So "1" corresponds to 10%, "4" to 40%, etc.

```
> Ticks:= zip(`=`, map(t-> (3*t-1)/2, ts), map(x-> sprintf("%1d", round(10*x)), ts));
```

$$Ticks := [-.35000000000000 = "1", -.20000000000000 = "2", -.05000000000000 = "3", .10000000000000 = "4", .25000000000000 = "5", .40000000000000 = "6", .55000000000000 = "7", .70000000000000 = "8", .85000000000000 = "9"]$$

Let's view the plot that we have so far.

```
> plotsetup(inline);
plot3d(1-x-y, x=0..1-y, y= 0..1
,style= wireframe
,gridstyle= triangular
,grid= [10,10]
,color= grey
,axes= normal
,orientation= [theta, phi]
,view= [axes_range $ 3]
,tickmarks= [Ticks $ 3]
,labels= [``$3]
,scaling= constrained
,projection= 1
);
```



Of course, it will appear better when it is bigger and plotted in a separate window. The above plot is just a proof-of-concept.

The grid lines in the above plot correspond to the x-y plane. We want the grid lines to correspond to the new axes. We need to take the lines x = constant, y= constant, and z= constant and project them into the triangle. So we need a transformation that projects objects from the x-y plane up into the plane z = 1-x-y. I throw out any values not in the first octant.

```
> Tr:= plottools[transform]
((x,y)-> map(p-> `if`(p>=0 and p<=1, p, undefined), [x,y,1-x-y])):
```

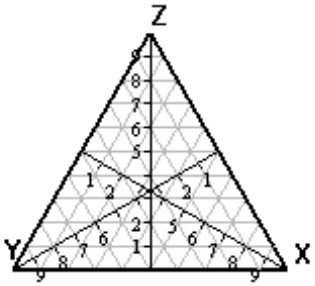
```
> Triangle:=
plots[display]
# Plot the grid lines in the x-y plane, including the lines
# corresponding to z=constant, and project them up.
# I chose to make the grid line values the same as the tickmark values,
# but you can choose to use greater or fewer grid lines if you want.
([Tr(plot([map(t-> [[0,t], [1-t,t]], ts)[] # y gridlines
,map(t-> [[0,1-t], [1-t,0]], ts)[] # z gridlines
,map(t-> [[t,0], [t,1-t]], ts)[] # x gridlines
]
,color= grey
)
)
,plots[spacecurve] # The three borders.
({[[0,0,1], [0,1,0]], [[0,1,0], [1,0,0]], [[1,0,0], [0,0,1]]}
,color= black
,thickness= 2
)

# Put axes labels at the ends of the axes. The coordinates to get the
# labels to appear at the right place outside the triangle were
# computed with painstaking trial-and-error.
,plots[textplot3d]
({[1.0,0,.05,` X`], [-.05,1.0,.05,`Y `], [-.07,-.03,1.0,` Z `]}
,color= black
)
]
,view= [axes_range $ 3]
,orientation= [theta, phi]
,axes=normal
,scaling= constrained
,labels= [`` $ 3]
,font= [HELVETICA,BOLD,10]
,axesfont= [TIMES,ROMAN,7]
,tickmarks= [Ticks $ 3]
,projection= 1
):
```

Let's look at what we have so far in a separate window.

```
> plotsetup(window); print(Triangle); plotsetup(inline);
```

The triangle plots give acceptable results, though somewhat small, when plotted inline.

> **print(Triangle);**



Now let's do a mixing problem. The following problem is from page 425 of *Applied Regression Analysis* , Norman R. Draper and Harry Smith (3rd ed., Wiley, New York, 1998).

First input the experimental design. These are the proportions of the three chemicals at which the response is measured.

> **X1:= [1, 0, 0, .5, .5, 0, .2, .3]:**

> **X2:= [0, 1, 0, .5, 0, .5, .6, .5]:**

> **X3:= [0, 0, 1, 0, .5, .5, .2, .2]:**

Make a quick check that each design point adds to 1.

> **X1+X2+X3;**

[1, 1, 1, 1.0, 1.0, 1.0, 1.0, 1.0]

Input the observed responses.

> **Y:= <40.9, 25.5, 28.6, 31.1, 24.9, 29.1, 27.0, 28.4>:**

Because of some chemical theory not mentioned, we have reason to believe that the response can be modeled by a quadratic function.

```
> Ypred:= x->
beta[1]*x[1]+beta[2]*x[2]+beta[3]*x[3]
+beta[4]*x[1]*x[2]+beta[5]*x[1]*x[3]+beta[6]*x[2]*x[3];
```

$$Ypred := x \rightarrow \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3$$

Because of the constraint x1+x2+x3 = 1, it is not necessary to include a constant term or squared terms in the above model. We'll see those terms appear later. See pages 414-415 of Draper & Smith for a complete explanation of that. In other words, if you make the substituton x3= 1-x1-x2, then the resulting model will have all the required terms.

We now do linear least squares on the data to compute the coefficients β . We need to construct the predictor matrix with columns including the cross product terms.

```
> X:= <<X1[]> | <X2[]> | <X3[]>
| <zip(`*`,X1,X2)[]> | <zip(`*`,X1,X3)[]>
| <zip(`*`,X2,X3)[]>
>:
```

```
> beta:= Y&/X:
```

So our prediction function is

```
> 'Y(x)' = map(evalf[5], Ypred(x));
```

$$Y(x) = 40.910\, x_1 + 25.432\, x_2 + 28.610\, x_3 - 8.2135\, x_1 x_2 - 39.339\, x_1 x_3 + 8.0016\, x_2 x_3$$

Apply the sum-to-one constraint to simplify that.

```
> Y2:= unapply(expand(subs(x[3]= 1-x[1]-x[2], Ypred(x))), x);
```

$$Y2 := x \rightarrow -27.039320883658\, x_1 + 4.8239062240466\, x_2 + 28.609993118667 + 23.124191443419\, x_1 x_2 + 39.3393208836580471\, x_1^2 - 8.00162304534561208\, x_2^2$$

There's the constant term and the squared terms that were absent from the model.

It's a little easier to work with if we make it a function of non-subscripted variables:

```
> Y2:= unapply(Y2([x,y]), x,y):
```

We want to take a contour plot of that function and project it into the triangle. Now we could just get a contour plot of this function with the contourplot command. However,

the function is quadratic and it can be solved explicitly for each contour level c. This gives far more accurate results and much greater control over the colors of the contours.

Since the function is quadratic, there is a plus and a minus branch for each contour (though some of the branches will not fall into three-ingredient space).

```
> f:= map(unapply, map(evalf, [solve(Y2(x,y)=c, y)]), c,x);
```

$$f := [(c, x) \rightarrow .30143298407768 + 1.4449688089762\, x + .49989858024201\, 10^{-17} \sqrt{.14671493630007\, 10^{36} - .10036499224887\, 10^{36}\, x + .28028779628056\, 10^{36}\, x^2 - .50010144033410\, 10^{34}\, c}, (c, x) \rightarrow .30143298407768 + 1.4449688089762\, x - .49989858024201\, 10^{-17} \sqrt{.14671493630007\, 10^{36} - .10036499224887\, 10^{36}\, x + .28028779628056\, 10^{36}\, x^2 - .50010144033410\, 10^{34}\, c}\,]$$

The huge coefficients under the square root make me worry about catastrophic cancellation. Why did Maple choose to factor out that very small coefficient? I can put the coefficient back under the square root with a bit of effort.

```
> disc, nondisc:=
selectremove(patmatch, f[1](c,x), a::numeric*b::And(polynom,Not(numeric))^(1/2));
```

$$disc, nondisc := .49989858024201\, 10^{-17} \sqrt{.14671493630007\, 10^{36} - .10036499224887\, 10^{36}\, x + .28028779628056\, 10^{36}\, x^2 - .50010144033410\, 10^{34}\, c}, .30143298407768 + 1.4449688089762\, x$$

```
> disc:= sqrt(op(1,disc)^2*op([2,1],disc));
```

$$disc := \sqrt{3.6663855790790 - 2.5081070101344\, x + 7.0043525232706\, x^2 - .12497464506050\, c}$$

```
> f:= [unapply(nondisc+disc, c,x), unapply(nondisc-disc, c,x)];
```

$$f := [(c, x) \rightarrow .30143298407768 + 1.4449688089762\, x + \sqrt{3.6663855790790 - 2.5081070101344\, x + 7.0043525232706\, x^2 - .12497464506050\, c}, (c, x) \rightarrow .30143298407768 + 1.4449688089762\, x - \sqrt{3.6663855790790 - 2.5081070101344\, x + 7.0043525232706\, x^2 - .12497464506050\, c}\,]$$

That looks like a much safer function to plot. (Actually, I did it both ways, and it did not make any difference)

If we solve for x as a function of y, plot these parametrically, and overlay with the plots of f, then the gaps caused by the separate branches will disappear.

```
> g:= map(unapply, map(evalf, [solve(Y2(x,y)= c, x)]), c,y);
```

$$g := [(c,y) \to .34366786558955 - .29390684592404\,y + .10167943701493\,10^{-17}\sqrt{-.58919780082008\,10^{36} - .31400025697957\,10^{36}\,y + .28028779628056\,10^{36}\,y^2 + .24587075552286\,10^{35}\,c}, (c,y) \to .34366786558955 - .29390684592404\,y - .10167943701493\,10^{-17}\sqrt{-.58919780082008\,10^{36} - .31400025697957\,10^{36}\,y + .28028779628056\,10^{36}\,y^2 + .24587075552286\,10^{35}\,c}]$$

> **disc, nondisc:=**
selectremove(patmatch, g[1](c,y), a::numeric*b::And(polynom,Not(numeric))^(1/2));

$$disc, nondisc := .10167943701493\,10^{-17}\sqrt{-.58919780082008\,10^{36} - .31400025697957\,10^{36}\,y + .28028779628056\,10^{36}\,y^2 + .24587075552286\,10^{35}\,c}, .34366786558955 - .29390684592404\,y$$

> **disc:= sqrt(op(1,disc)^2*op([2,1],disc));**

$$disc := \sqrt{-.60915439648789 - .32463569411020\,y + .28978136569512\,y^2 + .025419859253732\,c}$$

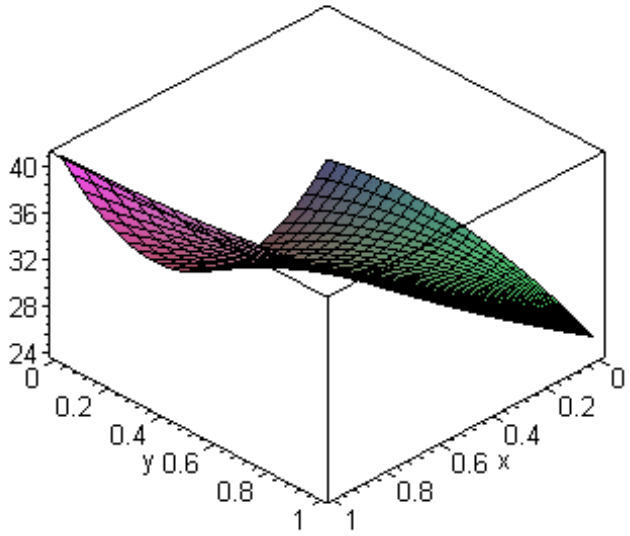
> **g:= [unapply(nondisc+disc, c,y), unapply(nondisc-disc, c,y)];**

$$g := [(c,y) \to .34366786558955 - .29390684592404\,y + \sqrt{-.60915439648789 - .32463569411020\,y + .28978136569512\,y^2 + .025419859253732\,c}, (c,y) \to .34366786558955 - .29390684592404\,y - \sqrt{-.60915439648789 - .32463569411020\,y + .28978136569512\,y^2 + .025419859253732\,c}]$$

It would be useful to know the range of the predicted reponse. I could use calculus to do this, but since I have Maple available, and since I only need an estimate, I'd like to show you a different way. We "read" the range from a 3d plot.

Note how I restrict the domain in the plot command.

> **plotsetup(inline); plot3d(Y2(x,y), x= 0..1-y, y= 0..1, axes= boxed);**



If you examine the plot and rotate it, you'll see that the z-range is roughly 24..41. Let's get Maple to compute that.

Extract the array of mesh values from the plot:

> **A:= op([1,1], %);**

$$A := \left[\begin{array}{l} 1..25 \text{ x } 1..25 \text{ x } 1..3 \text{ 3-D Array} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{array}\right]$$

The z-coordinates are the 3rd values in the 3rd dimension.

> **Z:= map(op, &LL A[[op([2,1],A)], [op([2,2],A)], 3])[]:**

> **Zmin:= min(Z); Zmax:= max(Z);**

$$Zmin := 23.9679218111874520$$

$$Zmax := 40.9099931186670176$$

Choose the number of contours that you want. Forty can be done in a reasonable amount of time. (In a future worksheet, I'll show how to get hundreds of contours in a reasonable time).

```
> ncontours:= 40:
```

Select evenly spaced contour levels.

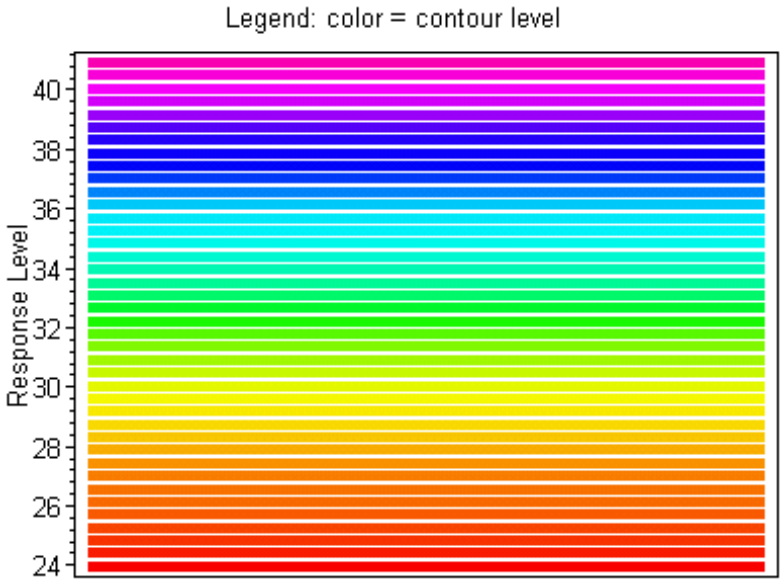
```
> contours:= [seq(Zmin+(Zmax-Zmin)*i/(ncontours-1), i= 0..ncontours-1)]:
```

Select a distinct color for each contour. I select my colors so that red corresponds to the lowest contour, and then they proceed through the color spectrum to violet. My function HSVspline gives a more evenly spaced and smoother distribution of colors, for the purpose of numerical encoding, than does Maple's default HSV. I will explain this in detail in a future worksheet.

```
> HSVspline:=
unapply(spline([seq(i/24, i= 0..24)]
,[0,.03,.05,.07,.08
,.1,.13,.15,.168
,.19,.23,.27,.34
,.41,.45,.48,.503
,.52,.59,.66,.68
,.73,.8,.84,.88
]
,x
)
,x):

> Colors:=
[seq(COLOR(HSV, HSVspline(i/(nops(contours)-1)), 1, 1), i= 0..nops(contours)-1)]:
```

Print a legend showing the correspondence between color and contour level.

```
> plotsetup(inline);
plot(map(t-> [[0,t], [1,t]], contours)
,color= Colors
,title= `Legend: color = contour level`
,labels= [``, `Response Level`]
,labeldirections= [HORIZONTAL, VERTICAL]
,thickness= 5
,xtickmarks= 0
,axes= boxed
);
```

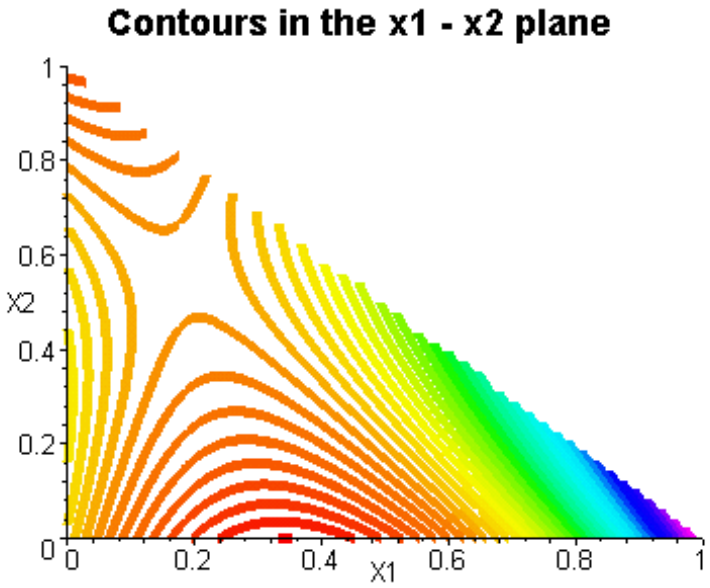


Plot the functions f and g at each contour level.

```
> P:= plot(map(c-> (map(y-> y(c,x), f)[]
,map(x-> [x(c,y), y, y= 0..1], g)[]
)
,contours
)
,x= 0..1
,color= map(C-> C$4, Colors)
,thickness= 5
):
```

Let's look at the contours in the x1-x2 plane. I cut off any portion of the plot that is above the line x1 + x2 = 1.

```
> plotsetup(inline);
plots[display]([plottools[transform]((x,y)-> `if`(x+y>1, [undefined$2], [x,y]))(P)]
,title= `Contours in the x1 - x2 plane`
,titlefont= [HELVETICA,BOLD,14]
,labels = ['X1', 'X2']
,view= [(0..1) $ 2]
,axes= frame
);
```



So the maximal respose is at x1 = 1, and the minimal response is at roughly x1 = .35, x2 = 0.

Project to the triangle. This next command may take about 30 seconds, or likely more on a computer slower than what I'm using right now. In a future worksheet, I'll show some ways to speed this up.

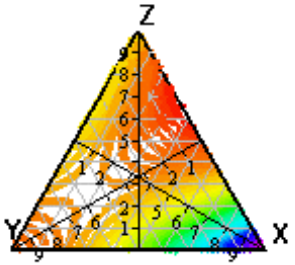
```
> Tri_contours:=
plots[display]
([Tr(P), Triangle]
,title = `Contours in the constrained x1,x2,x3 plane`
,titlefont= [HELVETICA,BOLD,14]
):
```

```
> plotsetup(window); print(Tri_contours); plotsetup(inline);
```

We might as well look at that inline also.

```
> plotsetup(inline); print(Tri_contours);
```

Contours in the constrained x1,x2,x3 plane



Examination of the contour plot makes it obvious that the predicted value at the centroid (1/3, 1/3, 1/3) will be between 27 and 28.

> **Ypred([1/3, 1/3, 1/3]);**

$$27.256175930100$$

Clearly, there is also a saddle point with a response between 27 and 28. Let's find that with calculus.

> **solve({seq(D[i](Y2)(sad[1],sad[2])=0, i= 1..2)});**

$$\{sad_2 = .56013900916555, sad_1 = .17903917612669\}$$

> **assign(%);**

> **sad[3]:= 1-sad[1]-sad[2];**

$$sad_3 := .26082181470776$$

> **Ypred(sad);**

$$27.540473277972$$

Let's test the significance of the regression with an F-test. The following procedure will return the p-value of the F-test for the *overall* regression.

```
> sig_of_regr:= proc(X::Matrix, Y::Vector)
local n,q,`Ypred'Y`;
n,q:= op(1,X);
`Ypred'Y`:= &T (X . Y&/X) . Y;
1 - stats[statevalf, cdf, fratio[q-1, n-q]]
((`Ypred'Y` - n*stats[describe, mean](&L Y)^2) # SS(reg)
/((&T Y . Y) - `Ypred'Y`) # SS(residual)
*(n-q)/(q-1)
)
end proc:

> sig_of_regr(X,Y);
```

.00786891562499

A p-value of less than 1% is pretty good considering that there were only 8 data points.

It is also possible to test individually the significance of the inclusion of each term. I will explore that in a future worksheet. In this problem, there is already some unmentioned theoretical reason for assuming the stated quadratic model. We also note that all of the coefficients are of the same order of magnitude, which is evidence that they are either all significant or all insignificant, and the F-test has already excluded the latter possibility.

It is always a good idea to examine the residuals when doing regression. The residuals are the differences between the predicted and measured values. In order for significance tests to be valid, we need the residuals to be normally distributed. Because of the Central Limit Theorem, this is roughly equivalent to saying that any deviations between the measured and predicted values are due to the sum of a variety of random influences, for example, errors inherent to the measuring process. We don't want there to be any functional relationship between the error and the point at which the measurement was made.

The following procedure produces two plots. The first plot has the sorted residuals on the horizontal axis, scaled by their standard deviation. On the vertical axis is the normal score of the position number of the sorted residual, slightly scaled. Thus it is a plot of the cumulative normal probability versus the residuals. The slight scaling is required because if we allowed the maximal residual to have a cumulative probability of 1, its normal score would be infinity, and we couldn't plot it. See the comments in the code and in the text block after the code for more discussion of this scaling. This plot is called a normal plot or a normal probability plot. If the distribution is normal, this plot should be close to a straight line. I also compute the linear correlation, skewness, and kurtosis. If the distribution is normal, the correlation should be very close to 1, the skewness close to 0, and the kurtosis close to 3. I put the least squares line on the plot. I balance the horizontal axis about the mean of the residuals (which is always exactly 0) so that skewness is easier to see. I think that my normal plot is a bit more useful than MINITAB's.

The second plot is simply the predicted value versus the residuals. This plot should appear completely random. Its correlation should be extremely close to 0.

```
> ResPlot:= proc(X::Matrix, Y::Vector)
local n, i, res, res_sorted, Ypred, zscore, s, x, y, res_range
,scalefn, scaletype, opts
;
Ypred:= X . Y&/X;
res:= Y - Ypred;
```

```
res_sorted:= sort(&L res);

# Scale by the standard deviation.
# Note: I consider the residuals to be an entire population rather than a sample;
# I am not sure which approach is more valid.
s:= stats[describe, standarddeviation[0]](res_sorted);
res_sorted:= map(x -> x/s, res_sorted);

# Compute the zero-centered range, and stretch it by 10% to unclutter the plot.
res_range:= 1.1*max(abs(res_sorted[1]), abs(res_sorted[-1]));

# There are several ways of scaling the zscores. By default, I choose the
# modified Kaplan-Meier estimate for complete samples, which is recommended
# by Draper & Smith (ibid, p. 71)
opts:= [args[3..-1]];
if not hasoption(opts, 'scale'={name, procedure}, 'scaletype', 'opts') then
scaletype:= `modified Kaplan-Meier`
fi;
if scaletype = 'BMDP' then scalefn:= (i,n)-> (i-1/3)/(n+1/3)
elif scaletype = `MINITAB default` then scalefn:= (i,n)-> (i-.375)/(n+.25)
elif scaletype = `modified Kaplan-Meier` then scalefn:= (i,n)-> (i-.5)/n
elif scaletype = `Herd-Johnson` then scalefn:= (i,n)-> i/(n+1)
elif scaletype::procedure then scalefn:= scaletype
else error `Unknown scale; use functional form`
fi;
n:= nops(res_sorted);
zscore:= [seq(stats[statevalf, icdf, normald](scalefn(i,n)), i= 1..n)];
print();
print
(plot([zip((x,y) -> [x,y], res_sorted, zscore)
,rhs(stats[fit, leastsquare[[x,y]]]([res_sorted, zscore]))
]
,x= -res_range..res_range
,'color'= ['red', 'black']
,'title'= `Normal plot of residuals`
,'labels'= [`standardized residuals`, 'nscore']
,'labeldirections'= ['HORIZONTAL', 'VERTICAL']
,'style'= ['point', 'line']
,'symbol'= 'box'
,'symbolsize'= 15
,'axes'= 'normal'
,opts[]
)
);
print(`Linear correlation of normal plot `
```

```
= stats[describe, linearcorrelation](res_sorted, zscore)
);
print(`Skewness and kurtosis of residuals `
= evalf[5](stats[describe, skewness[0]](res_sorted))
,evalf[5](stats[describe, kurtosis[0]](res_sorted))
);
print(`\n`);
print
(plot(&LL <res | Ypred>
,-res_range*s..res_range*s
,'title' = `Predicted response vs resdiuals`
,'labels'= ['residual', 'response']
,'labeldirections'= ['HORIZONTAL', 'VERTICAL']
,'style'= 'point'
,'symbol'= 'box'
,'symbolsize'= 15
,'axes'= 'normal'
,opts[]
)
);
print(`Linear correlation of the response-vs-residual plot `
= stats[describe, linearcorrelation](&L res, &L Ypred)
);
end proc:
```

The possible scales in the next command are `modified Kaplan-Meier`, 'BMDP', `MINITAB default`, and `Herd-Johnson`. The default is `modified Kaplan-Meier`. You can also input your own scale as a function f(i.n) of positive integer arguments with i <= n that returns a number in the open interval (0,1). For fixed n, it should be a strictly increasing function of i. The purpose is to approximate the true cumulative probability of the finite set of sorted values, which is i/n with i the position number and n the total number, but avoid the values 0 and 1. The precoded scales are

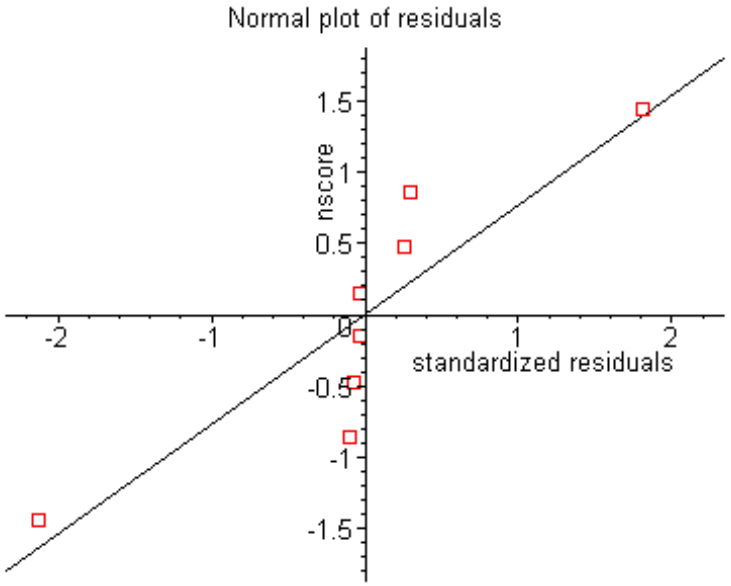
`modified Kaplan-Meier` (i-0.5)/n (my default)

`BMDP` (i-1/3)/(n+1/3)

`MINITAB default` (i-3/8)/(n+1/4)

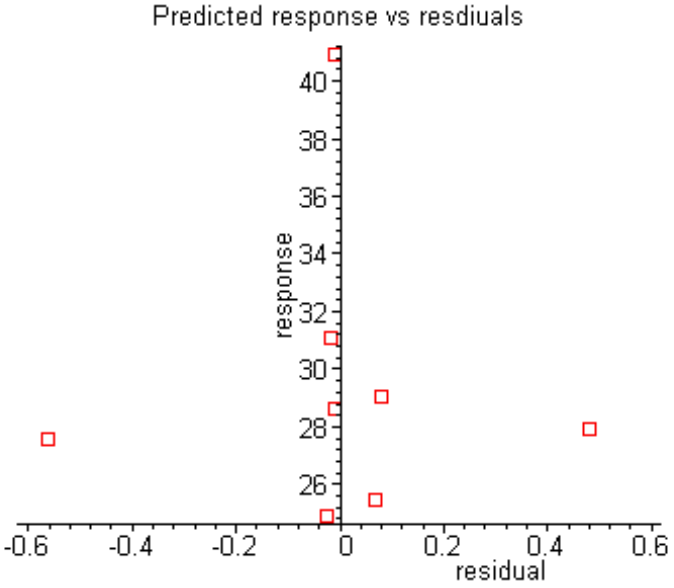
`Herd-Johnson` i/(n+1).

```
> plotsetup(inline); ResPlot(X,Y, scale= `MINITAB default`);
```



Linear correlation of normal plot = .88229647004220

Skewness and kurtosis of residuals = -.45425, 3.9301



$$\textit{Linear correlation of the response-vs-residual plot} = .12015828720326\ 10^{-13}$$

There are two outliers and the other 6 residuals are very close to zero. The two outliers are close to 2 standard deviations from the mean of the residuals (seen from the horizontal axis of the first plot), so they are not too bad. But it is likely that they had a major impact on the values of the coefficients. On the other hand, we see from the second plot that the most deviant residual is only about 2% of its corresponding predicted value.

If you find the position of the axes annoying, you could say ResPlot(X,Y, axes= frame). I find that the centered axes let me quickly assess the balance of the residuals.

Let's find the those outlying residuals. The following procedure will sort the residuals are find the positions in the original data to which they correspond.

```
> SortRes:= proc(X::Matrix, Y::Vector)
local res, i;
res:= &L (Y - X . Y&/X);
res:= [seq([i,res[i]], i= 1..nops(res))];
Matrix
(sort(res, (p,q) -> evalb(abs(p[2]) > abs(q[2])))
[1..`if`(nargs=3, args[3], op([1,1],X))]
)
end proc:
```

```
> SortRes(X,Y);
```

$$\begin{bmatrix} 7 & -.564364188595970262 \\ 8 & .481088199701023456 \\ 6 & .0784595306456594700 \\ 2 & .0677237026315218316 \\ 5 & -.0251628977528781661 \\ 4 & -.0177581092945366948 \\ 3 & -.00999311866740271172 \\ 1 & -.00999311866737429000 \end{bmatrix}$$

So the outliers correspond to the 7th and 8th design points.

My procedure SortRes can also just list the top few residuals

```
> SortRes(X,Y,2);
```

$$\begin{bmatrix} 7 & -.564364188595970262 \\ 8 & .481088199701023456 \end{bmatrix}$$

Select the corresponding design points from the data X.

> **outliers:= &LL X[&L %[[1..-1], 1], [1..3]];**

$$outliers := [[.2, .6, .2], [.3, .5, .2]]$$

It is interesting to note that those two points are very close to the saddle point. But also note that these are the only two design points in the interior of the triangle, and they are close to each other. They are the only two that use all three ingredients. It might be reasonable to suspect that the error in the measuring equipment changes when all three ingredients are used, or that some other factor comes into play. These points need to be investigated further.

Let's redo the regression without the outliers, just to see "what if". You shouldn't disregard outliers without some good physical reason, but it is trivial to redo the regression without them.

> **beta2:= Y[1..6] &/ X[[1..6],[1..-1]];**

$$\beta 2 := \begin{bmatrix} 40.8999999999999986 \\ 25.4999999999999966 \\ 28.6000000000000050 \\ -8.3999999999998080 \\ -39.4000000000000057 \\ 8.20000000000001526 \end{bmatrix}$$

Compare with the original β .

> **beta;**

$$\begin{bmatrix} 40.9099931186673728 \\ 25.4322762973684782 \\ 28.6099931186674042 \\ -8.21350639489354783 \\ -39.3393208836580471 \\ 8.00162304534561208 \end{bmatrix}$$

The removal of the outliers did not make a big difference, which provides some more evidence that the original regression was significant. But we cannot analyze this any further (by means that I know of), because in this latter regression we are finding 6 coefficients with only 6 data points. The residuals are necessarily 0 (up to round-off error).

```
> &L (Y - X.beta2)[1..6];
```

$$[0., .355271367880050094\ 10^{-14}, -.355271367880050094\ 10^{-14}, 0., 0., -.355271367880050094\ 10^{-14}]$$

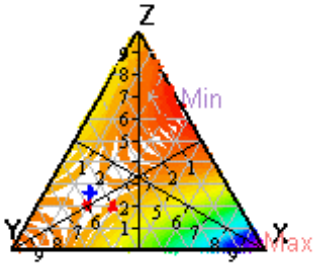
Let's add the saddle point, outliers, and extrema to the contour plot.

```
> plotsetup(window);
plots[display]
([Tri_contours
,plots[pointplot3d](outliers, symbol= CROSS, symbolsize= 15, color= red)
,plots[pointplot3d](&L sad, symbol= CIRCLE, color= blue)
,plots[textplot3d]([[1,0,0,`Max`], [.35,0,.65, `Min`]])
]
,title= `Contours, saddle, outliers, and extrema`
,titlefont= [HELVETICA,BOLD,14]
);
plotsetup(inline);
```

Look at it inline also.

```
> %;
```

Contours, saddle, outliers, and extrema



We recommend that measurements be taken at more and more widely separated interior points before we have strong confidence in the prediction function.

As you may have guessed, it is also possible to do a four-ingredient mixing problem and represent the results in a 3-dimensional regular tetrahedron. Since there are no 4d plotting commands, we would have to explicitly draw the axes and tickmarks and define a projection matrix from 4-space into the tetrahedron. We could also take triangular slices of the tetrahedron and represent the fourth dimension as an animation of these slices. I will explore these in a future worksheet.

>