The Geometry of Trend (Regression) Analysis **Draft 04-16**

[rob rucker 2008]

The purpose of this tutorial is to enable you to do trend analysis, emphasizing the *geometry* of the variables under consideration. To me, trend analysis means describing how variables influence each other both mathematically as well as geometrically. More specifically, I'll show you how one variable, called by various names such as: the response, observation, dependent, or outcome variable, can be predicted/estimated from others (called predictors). Once the prediction is made, we will determine just how good that estimate is. Along the way you will see how *variables* can be represented by vectors, directed line segments in space, and how the angles between those vectors can be calculated. This will let us visualize correlation coefficients, which are measures of the goodness of your prediction. This visualization aspect of this analysis, beyond prediction equations, allows determination of the *configuration* of the represented variables, that is, their relative lengths and the angles between them. This provides both an interpretation of the contributions of various variables, as well as a reality check on any proposed model.
My mantra is: *if I can represent something then I can critique it, and geometry is crucial for representation.*
Note : If you have only a couple of minutes for this, let me suggest you read just the snippets below, either *Micro-Cut to the Chase*, or the more pedestrian *Cut to the Chase*. Additionally, there is a simpler and earlier example of trend analysis on this site called: *Business Trend Analysis* that might be a better starting point. (I have a little more description of that tutorial in a following section).

- **Suggested Background for this Tutorial**

Mostly, reading this tutorial just takes interest and a resistance to being overwhelmed, although a little math background wouldn't hurt either. I am assuming that you know how to add and subtract Vectors, how to do a Dot product, what a Vector Space means, how it' s represented, and, how to project vectors onto Vector Spaces. If these are hazy ideas to you, check out other tutorials on this site, such as *Trigonometry Basics* and *Vector Arithmetic and Vector Operations*. In any event, at least just *check out the diagrams* since those carry almost all of the meaning of the regression approach.

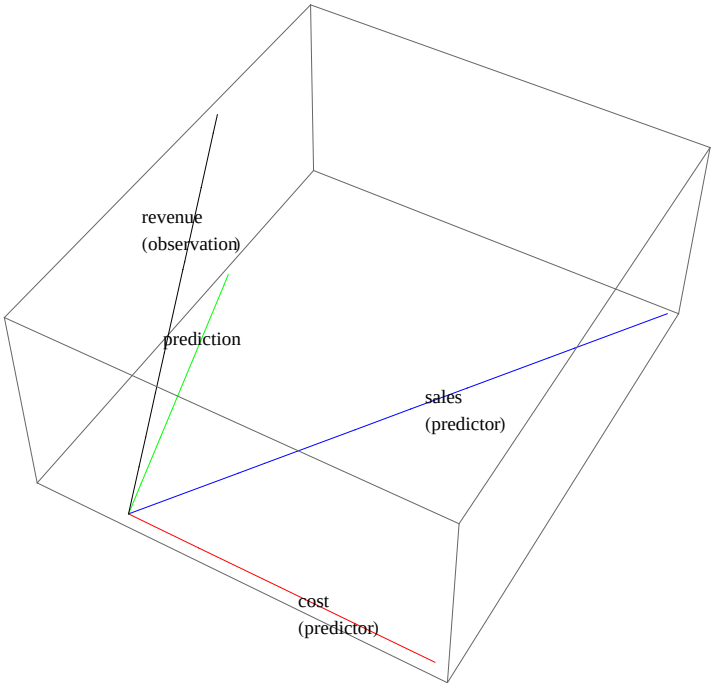
Micro-Cut to the Chase - (An Overview for People with No Time)

Calculate your regression equation by a geometric analysis of representative vectors of your variables - there, wasn't that easy! See the picture "End Goal Diagram" below for a picture of the goal of the geometric approach.
O.k, a little more detail: Distinguish one variable as the one to be predicted, call it the *dependent* variable (maybe revenue?), and the others as the *predictor* variables, (sales and cost)?
Represent your variables by columns of their data, that is, by *vectors*.Then, transform the data of each vector by subtracting off its mean, resulting in what is called 'centered data'. All vectors in this tutorial will be assumed to be centered. That is a requirement for the geometric interpretations to come. Note, you often are after a little different insight, and that is: how do I interpret the predictor variables and what are their inter-relationships. That too falls out as a consequence of this geometric approach.
Write the dependent vector as a sum of multiples of the predictor vectors. Find the best multipliers of the predictor vectors by picking the multipliers that bring the combination (vector sum) of predictor variables closest to the dependent vector. Call that best combination of the predictor vectors the *prediction* vector. The closeness of the tip of the prediction vector to the tip of the observation vector is a measure of how good the prediction is. *Closer is better*! An equivalent measure (easier to calculate) of the goodness of fit of the prediction vector to the dependent vector is the angle between the two. Small angle -> good prediction, large angle -> poor prediction. The Cosine of this angle is called the Correlation Coefficient. Cosines of angles run from -1 to +1. Small angles are close to -1 or +1 while large angles are closer to 0. So correlation coefficients near -1 or +1 means 'good fit' while coefficients near 0 mean 'poor fit'. Solving the vector equations will result in the determination of the regression equation coefficients. That's it.
[Beware - all of the above depends on the fact of the dependent vector being linearly related to the predictor variables.(You may have to transform you data set in order to approximate linearity). Consult your domain expert as well as your local captive statistician as to the reasonableness of this assumption!]

The End Goal Diagram - showing the configuration of the variables

This diagram shows the interrelation of the Sales, Cost, Prediction and Revenue variables represented as vectors. This is the geometric view of the regression equations. The angles between each vector have been calculated and displayed relative to each other. For example, I have calculated that the Sales and Cost vectors have a 55 degree angle between them. Between Revenue and the Prediction vector there is a 34 degree angle which in turn tells me the multiple correlation coefficient = 0.83. Having such pictures gives me some confidence in my interpretations of results, and gives my clients a warm fuzzy as well! These calculations, and extensive discussion of their meanings, appear later in this tutorial.

Revenue vector predicted from Sales and Cost vectors (all data is centered).
Relative lengths (variability) of cost and sales vectors are shown.
The Cosine of the angle between revenue and prediction vectors is
the multiple correlation coefficient



■ Cut to the Chase - (An Overview for People with a Little More Time)

If you don' t have time to follow all the details presented later in this tutorial, here is a synopsis of what I will cover that will give you some idea of the sequence you would normally go through to do a trend analysis.(I have repeated this sequence of phases using numeric data in the next section). See "The End Goal Diagram" diagram above for the geometry we are working toward. Note: rather than a strict prediction of one variable by others, you are often after a little different insight, and that is: *how do I interpret the predictor variables and their inter-relationships*. That too falls out as a consequence of this geometric approach, where you can measure the length and angles of vectors representing variables, that is, their *configuration*.

Phase 0. This is the where you decide what variable(s) need to be predicted and what variables you are going to use to make those predictions. Determining these variables depends on the specific domain and specific situation you are working in. For each of these variables you will need a list of values, collected under the same conditions.

Phase 1. Suppose that you have decided to find the trend of a particular variable, that is, to *predict* a variable, say Revenue. Suppose you can't directly control Revenue, but can observe it. Assume you have reason to believe that Revenue *depends* on, or can be *controlled*, by other variables, say Sales and Cost. Suppose further that Sales and Cost are variables whose values either you know, can observe, or are easier to measure. So, what you have available is your set of observations of the variable Revenue, along with matching data sets of what are called the *predictor* variables, Sales and Cost.

Phase 2. To predict Revenue you can now write a (linear) equation. This is a major assumption that will require some additional investigation to determine if there really is a linear relationship between your variables (or at least an approximate one). This is where various transformations of you data (log, square root, ArcSin. . .) come in, to try to make the relations linear. So, if you want to go ahead, then the next thing to do is to write down a tentative *linear* equation as below:

```
prediction = b1 * Sales + b2 * Cost; (* this scaled sum of the two variable,
Sales, and Cost,  this  is the prediction variable,
which  predicts revenue when you plug in specific values for Sales and Cost *)
```

where b1, and b2 are unknown real numbers, but you *can* calculate them from your lists. (How you could do this is shown in the tutorial below, but for now just assume you could). Now that you have b1, and b2, you can plug in any Sales and Cost values into the equation above and get a resultant prediction of a Revenue value. So now you can *predict* Revenue, given Sales and Cost values. Note that you actually *get b1 and b2 by solving a vector equation* as described below.

Phase 3. How good is your prediction? Now, to actually find b1 and b2 above I did have to solve some equations. The approach I use is to write vector equations involving the variables and then use geometric reasoning to derive the equations. In text books you will find formulas that reach equal conclusions following an all-algebraic approach.

To check out how good my prediction is, I look again at the vector equation represented by the prediction equation above. There are a couple of ways to answer the "goodness of fit" question. The simplest way is to calculate the angle between two vectors, the Prediction vector and the Revenue vector.

That angle between the observed Revenue vector and the Prediction vector will tell you how close they are to each other. That closeness measures the strength of their *correlation*. A small angle, the closer to zero the better, means a good prediction, while a 90 degree angle means your predictors won't help at all to predict the observed Revenue values. The cosine of this angle between the prediction vector and the observation vector is called the *Correlation Coefficient*, often written as 'r'. Cosines run from +1 to -1. Small angles (close predictions) correspond to cosines near +1 or -1 while large angles (poor predictions) correspond to cosines near zero.

Further, the square of the correlation coefficient is often calculated and is called the coefficient of determination, R^2. This gives you another measure of the 'goodness of fit'. It can be interpreted as the proportion of total variability of the dependent vector accounted for by the prediction. For a correlation of, say, 0.82, which corresponds to an angle of about 34 degrees, then $R^2 = 67\,\%$. (That is the angle shown in the End Goal Diagram above).

That's not a great correlation, but its' lots better than just using the average of the revenue values to predict other revenue values! It really is that simple!

Cut to the Chase Summary

Going through the three phases above yields a trend equation (also called a regression equation) that allows you to predict one variable from others. In addition, looking at the vector equivalent of the regression equation allows calculating the cosine of the angle between the dependent vector and the prediction vector. That gives you a measure of the *goodness of fit,(* how good your prediction is*),* called the correlation coefficient. Representing your variables as vectors also allows a visual check on the feasibility of your proposed model as well as the spatial interrelationships of all the variables. (I think about this perspective as a 'sanity' check on any results I get). {Keep in mind that this description applies when you work with centered data, and that the data sets are linearly related).]

A Simpler and Easier Example of the Regression Approach

You might want to start with an earlier tutorial on the this site called *Business Trend Analysis* that plotted the revenues from the training department of a consulting firm, against the month of occurrence of that revenue. There were two variables involved there, months, the predictor variable and revenue, the observation/dependent variable. This would be called a Bivariate Regression where "Bivariate" means two variables. The result of that tutorial was to show how to calculate the best line, the trend line, that predicted revenues from the month variable. That tutorial is an example of the regression approach in a very simple setting. (This tutorial also illustrates how a *scatter plot* is generated).

A More Balanced Picture - Two Views of Your Data Sets

If this is your only introduction to regression analysis, I better back up a moment and point out that there are two complementary ways to graphically represent data that you collect. I have emphasized one way, called plotting in *Subject* space , but there is an equally important approach that is called plotting in *Variable* space. (This latter approach is usually the only one presented since it is the perspective that leads to *scatter plots*). The word "Subject" shouldn't be interpreted too narrowly. In the social science literature this is often more specifically called *Person* space reflecting the usual measurement context in which measurements are done on people. In engineering, this notion of a subject might refer to a turbine whose parameters were being measured, in agriculture the 'subject' space could be some plant or animal and its response to various treatments, and in business, I might characterize my subject space as a 'situation' whose features I am recording.

To summarize the two views : in statistics, *Subjects are plotted in **Variable** space and Variables are plotted in **Subject** space.*

The fact that I am emphasizing one way to view multivariate data doesn't diminish the importance of the complementary view. Both are required in a professional engagement.
Let me illustrate the two views with the small table of data below. These two views can conveniently be illustrated by reading the table either by rows, or reading it by columns. Ok, look at the table below where the three "subjects" are Jo, Kim, and Zak. I collected data on these three subjects and interpreted "Y" as the dependent variable and "X1" as the independent variable. That is, I might be trying to predict "Y" by using "X1". (I have embedded the *Mathematica* 6.0 code right in this document since that allows me to calculate, diagram, and write, all in one context, besides- you might be motivated to get *Mathematica* and do your own calculations).

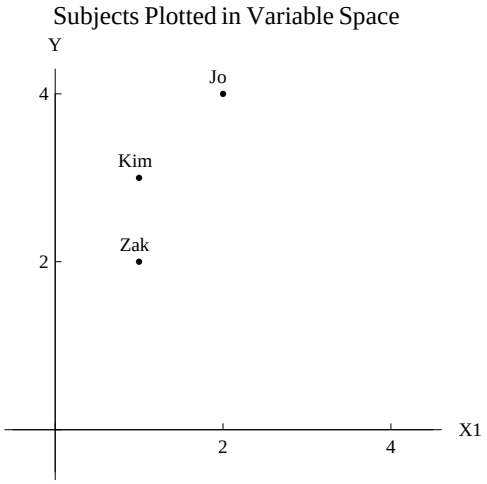
```
Grid[{{" ", "Y", "X1"}, {"Jo", 2, 4}, {"Kim", 1, 3}, {"Zak", 1, 2}},
 Alignment → Right, Frame → All]
```

	Y	X1
Jo	2	4
Kim	1	3
Zak	1	2

Plot the Subjects in Variable Space (this is the scatter plot perspective)

Reading across the table, and plotting the data by rows, means that the axes of the plots are the variables, and the points in space, represented by the rows of data, are the coordinates of each subject. This leads to what is called a scatter - plot, a textbook favorite. Notice that the connection between the X1 ,Y variables is implicit, and not easily seen.
The emphasis is on how the *subjects* interrelate. From this perspective you could detect outliers and could fit a line to the data if that made sense.

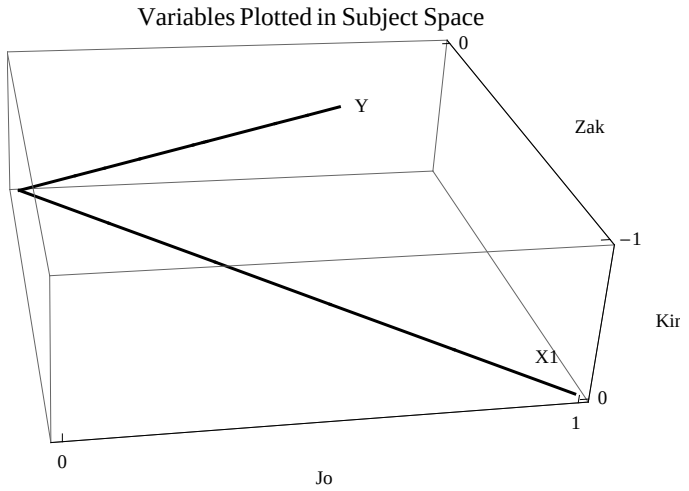
```
Graphics[{Line[{{-.5, 0}, {4.5, 0}}], Line[{{0, -.5}, {0, 4}}],
  Text["Jo  ", {2, 4.2}], Text["Kim  ", {1, 3.2}], Text["Zak  ", {1, 2.2}]},
 Axes → True, AxesLabel → {"X1", "Y"}, Ticks -> {{0, 2, 4}, {0, 2, 4}},
 PlotLabel → " Subjects Plotted in Variable Space",
 Epilog → Map[Point, {{2., 4}, {1., 3}, {1., 2}}]]
```



■ Plot the Variables in Subject Space (this is the variables-as-vectors approach)

[This is the perspective emphasized in this tutorial]. Reading down the table by columns leads to a different picture, where now the axes are the Subjects, and the points in space are the pure variables, "Y" and "X1". (Don' t be discourages if these two perspectives require some mind bending, that' s good for you)! That is, one point in subject 3- space is {2,1,1} representing the "Y" values of the three subjects. The point {4,3,2} represents the pure variable "X1", which is composed of the X1 components of the three subjects. This perspective lets me *see the relation between variables Y and X1*, such as their relative lengths and more specifically, the angle between them tells me the correlation between them. The Cosine of this angle *is* the correlation coefficient (if the data was centered, which will be discussed later).

```
Graphics3D[{Thickness[0.005],
  Line[{{0, 0, 0}, {2 / 3, -1 / 3, -1 / 3}}], Line[{{0, 0, 0}, {1, 0, -1}}],
  Text["Y", {2 / 3 + .05, -1 / 3, -1 / 3}], Text["X1", {1 - 0.07, 0 - .09, -1}]
   }, Axes → True, AxesLabel → {"Jo", "  Kim", "  Zak  "},
 PlotLabel → "Variables Plotted in Subject Space",
 Ticks -> {{-2, -1, 0, 1, 2}, {-2, -1, 0, 1, 2}, {-2, -1, 0, 1, 2}}]
```



Just to complete this, let me calculate the correlation between the Y and X1 variables. Correlation **is** the Cosine of the angle between them. I will do this explicitly by using the Dot product, scaled by the lengths of the Y and X1 vectors. 'Norm'.computes the length of a vector. First I must center the data by subtracting off means as follows.

```
centeredX1 = {4, 3, 2.} - Mean[{4, 3, 2}]
```

```
{1, 0, -1.}
```

```
centeredY = {2, 1, 1} - Mean[{2, 1, 1.}]
```

```
{0.666667, -0.333333, -0.333333}
```

```
CosineBetweenYX1 = Dot[centeredX1, centeredY] / (Norm[centeredX1] * Norm[centeredY])
```

```
0.866025
```

```
Angle = ArcCos[CosineBetweenYX1] * 180 / Pi
```

```
30.
```

So, the correlation between Y and X1 is 0.866 which says that the angle between them is 30 degrees - not bad for just three values!

Back to the Tutorial

O.k., to continue. The technical term for what I am about to discuss with you is called Multiple Regression, but don't worry, the geometric approach you have just seen, *generalizes to any dimension* and will enable you to understand and calculate predictions and correlations (which is what Multiple Regression is all about) using mostly just common intuition.(plus just a bit of math!). My approach here is enthusiastically geometric and the alternative ways to approach multivariate statistics, via matrices or computer packages are not emphasized. However, while not emphasizing calculations, I will show you how you *could* calculate all the required parameters, using just a bit of algebra, plus a lot of geometry. I do admit to using a general math package though, called *Mathematica.* I haven't used it's built in statistical packages, except to verify the geometry based calculations done here.

There are two major examples presented in this tutorial. The first is a made-up example showing how revenue could be predicted from sales and cost information. The second example uses real data to predict the *eruption schedule* of the YellowStone park geyser named Old Faithful. This example uses what I would call a *stepwise* approach, first trying just one predictor, and then adding a second predictor to try and get a better *prediction*.

For this tutorial, I am going to tackle the task of trend analysis in two parts: First: showing how to predict one variable in terms of one or more others, and second, calculating how good that prediction is.
I am going to approach this from the perspective that you have columns of data representing observations of each of the variables. I will interpret those columns of data as math objects called *vectors*.
Once the data is in vector format, a vector equation can be written that predicts one variable from the others, and secondly, the prediction can be assessed as to how good it is. All this can be done geometrically, as you will see, just using some basic ideas of vector operations and vector spaces.

Note : This document was produced using the math package called Mathematica. In its' native format, this document is live, meaning that any mathematics in the document is executable. For presentation on the web though, I converted this document to a pdf format.

Example 1: Revenue Depends on Sales and Cost

Phase 0. The hardest part of trend analysis is to decide what variable(s) you want to predict, and what variables you want to predict with, that is, what will be the *predictor* variables. For the example here, suppose I want to predict Revenue, as well as a tentative set of variables I am going to use as predictors, Sales, and Cost.

Phase 1. The deliverable from this first phase is a *prediction equation* that expresses the prediction of the observed variable as a linear combination of the chosen predictor variables. That is, suppose I am Director of Marketing and am trying to predict next years R*evenue* based on two other variables, say, *Sales*, and *Cost*. Further, suppose I have numeric data on R*evenue*, *Sales*, and C*ost* for the last 4 years.
Phase one of the trend analysis would result in my developing an equation like the one below. Note that this is called a linear equation since each variable is only multiplied by a constant and components like the square of a variable are not allowed. (There is a way to handle these other types of equations, called non-linear regression, but I won't cover that). b1 and b2 are constants to be determined. Once I knowb1 and b2 though, substituting in values for Sales and Cost would give me a value that would approximate Revenue.

prediction = b1 * Sales + b2 * Cost

Anticipating the geometry insight to come , let me recast the original raw data in an equivalent way, as a vector equation. Suppose the actual variables were the lists (vectors) below (suitably scaled). By scaled, I mean that a value of "4" might stand for $40,000. The reader may recall that these are similar to the numbers from the tutorial *Business Trend Analysis* although now, I am using two new variables to predict Revenue.

```
rawrevenue = {4, 3, 7, 8, 9};(* observed yearly revenue data (scaled) *)

rawsales = {1.3, 1.7, 1.6, 3, 5};(* observed yearly sales(scaled) *)

rawcost = {2, 4, 3, 3, 4};(* observed yearly costs or a surrogate for cost (scaled) *)

Join[{{"rawrevenue", "rawsales", "rawcost"}},
  Transpose[{rawrevenue, rawsales, rawcost}]] // Grid
```

rawrevenue	rawsales	rawcost
4	1.3	2
3	1.7	4
7	1.6	3
8	3	3
9	5	4

All Subsequent Analysis Requires that these Vectors be Centered. Subtracting off the means results in centered vectors.

```
Mean[rawrevenue] // N(* the 'N' converts numbers to decimal format*)
```

```
6.2
```

```
revenue = rawrevenue - Mean[rawrevenue] // N
  (*31/5 was   subtracted from each  component of the rawrevenue vector*)
```

```
{-2.2, -3.2, 0.8, 1.8, 2.8}
```

```
sales = rawsales - Mean[rawsales];(*center the rawsales vector *)

cost = rawcost - Mean[rawcost] // N;(*center the rawcost vector *)

Grid[Join[{{"revenue", "sales", "cost"}}, Transpose[{revenue, sales, cost}]],
 Alignment → Right]
```

revenue	sales	cost
-2.2	-1.22	-1.2
-3.2	-0.82	0.8
0.8	-0.92	-0.2
1.8	0.48	-0.2
2.8	2.48	0.8

■ Initial Geometric Insight into MultiColinearity

Right off the bat I can tell how close together my two predictor variables are. In this case the two vectors Sales and Cost are 55.8 degrees apart (as actually shown in the diagram above). How did I know that?

If there were only 3 component for each of these vectors, you would be in 3-Dimensions and could actually measure this angle by drawing the directions of the two vectors and using a protractor to measure the angle between them! Since that' s way too easy, math procedures have been developed to calculate this angle. To be fair to the mathematicians though, the vectors above are in 5-dimensional space, which is a little harder to visualize. So, we use some handy tools that work in any dimension, namely, the Dot product between two vectors. (See the tutorial on this site called *Trigonometry Basics* for background as well as the tutorial *Vector Arithmetic and Vector Operations.*).

That is, to find the angle between two vectors, I first find the cosine of the angle between the vectors using the Dot product, and from that, get the angle. Ok, here's the calculation. By definition, the cosine of the angle between two vectors is their Dot product divided by their lengths.

Note that the length of a vector is the Square root of its Dot product with itself. So, for example, the length of the *sales* vector is Sqrt[sales . sales]. Note: the Dot product is represented by a "." between two vectors.

```
cosineOfSomeAngle = (sales . cost) / (Sqrt[sales.sales] * Sqrt[cost.cost])
(*This cosine has additional meaning as
 the correlation coefficient of these two vectors *)
```

```
0.561729
```

What angle has a cosine of 0.5617? I'll use a built in *Mathematica* function that finds the angle with that given cosine. Since the answer is in radians, I convert that to degrees by multiplying by 180 / Pi

```
angle = ArcCos[cosineOfSomeAngle] *
  180 / Pi (* ok, the angle with cosine 0.5617 is 55.82 degrees *)
```

```
55.8245
```

For the future, I will use a built in function called *Correlation*[] that will calculate the cosine of the angle between (centered) variables directly.

```
Correlation[sales, cost]
(* this  built in function calculates the cosine between centered vectors,
compare this with with the equivalent Dot product  method above *)
```

```
0.561729
```

Question: What is the significance of a 55.8 degree separation between the Sales and Cost vectors? Since closeness is measured by the angle between the two variables considered as vectors, 55.8 degrees means they are fairly well separated and I would likely try to use *both* as my predictors. If , on the other hand, the angle had been very small between them, say 10 degrees or less, then I would say that they are highly correlated and, I can most likely use just *one* of them for my predictions and not bother with the other. When the predictor vectors are lying on top of each other, it is called multicollinearityand is a bad thing when not recognized!. *The geometry is a way to see this early on and make adjustments.*

There are some other gotcha's with vectors in space that the geometry helps with. Drawing out vector configurations may show a seemingly useless vector actually supplies a crucial additional dimensional direction that helps the prediction vector get closer to the dependent vector.

Along these lines, I can actually figure the angles between all of the vectors and so work out what configurations are actually possible. The major danger of regression analysis is just this closeness among predictors which screws up interpretation of results.(-Here is where relying on the numbers produced by statistics package may lead you astray)
Let me repeat : Given the possible calculation of the angles between vectors, you/I could actually construct stick figure models showing the possible orientations of the four vectors : sales, cost, revenue, and prediction. (That was the basis for the construction of the "End Goal Diagram" shown at the start of this doucment.
Note, I don' t yet have the prediction vector, but when I do get it, I can find all of its inter-angle relationships as well. Remember, the correlation coefficient is the Cosine of the angle between vectors, so to get the actual angle, I need to find the angle with that cosine. That' s the ArcCos function (read as "the angles whose cosine is. . ".

■ Initial Correlations Among the Vector So Far Given

```
corrsalescost = Correlation[sales, cost]
 (* this  built in function calculates the cosine between centered vectors *)
```

```
0.561729
```

```
angelBetweenSalesCost = ArcCos[corrsalescost] * 180. / Pi
```

```
55.8245
```

```
corrrevenuesales = Correlation[revenue, sales](*correlation between revenue and sales*)
```

```
0.780489
```

```
angleBetweenRevenueSales = ArcCos[corrrevenuesales] * 180. / Pi
```

```
38.6946
```

```
corrrevenuecost =
 Correlation[revenue, cost] // N (*correlation between revenue and cost vectors *)
```

```
0.207791
```

```
angleBetweenRevenueCost = ArcCos[corrrevenuecost] * 180. / Pi
```

```
78.0071
```

■ Perpendicularity Leads to Least Squares Equations and Therefore, Best Estimates

Here' s where we stand right now. In the matrix below, the left hand column of numbers are the centered *revenue* values. The right hand side is a column of numbers given by multiples of b1 and b2 applied to the sales and cost numbers, that is, the *prediction* vector. Since there are only two parameters, b1 and b2, and 5 equations, there is no way these quantities on both sides of the equal sign can all be equal. What I can do though, is make them match as closely as possible by adjusting b1 and b2. That process is called a *Least Squares Fit* and is usually how the adjustment is described. Another, equivalent perspective, is the one adopted here. In the geometric approach I write the left and right sides as vectors in space and adjust b1 and b2 to get their tips as close together as possible. In the end, both perspectives lead to the same equations, which lead to the same b1 and b2 solutions.

```
Clear[b1, b2]
```

```
prediction = b1 * sales + b2 * cost;
Transpose[Join[{revenue}, {{"=", "=", "=", "=", "="}}, {prediction}]] // MatrixForm
```

$$\begin{pmatrix} -2.2 & = & -1.22\,b1 - 1.2\,b2 \\ -3.2 & = & -0.82\,b1 + 0.8\,b2 \\ 0.8 & = & -0.92\,b1 - 0.2\,b2 \\ 1.8 & = & 0.48\,b1 - 0.2\,b2 \\ 2.8 & = & 2.48\,b1 + 0.8\,b2 \end{pmatrix}$$

Here is the (centered) form of the prediction vector I will use to estimate/predict revenue. At this point I don't know b1 or b2 but will find them using geometric insight and a little algebra.

```
prediction = b1 * sales + b2 * cost
```

```
{-1.22 b1 - 1.2 b2, -0.82 b1 + 0.8 b2, -0.92 b1 - 0.2 b2, 0.48 b1 - 0.2 b2, 2.48 b1 + 0.8 b2}
```

Now Calculate the b1 and b2 that will minimize the error between prediction and the dependent vector

The b1 and b2 values are calculated so that the tip of the prediction vector = b1 * sales + b2 * cost, is as close as possible to the tip of the observation vector. The difference between the tips is a vector itself called the *error* vector. The "as close as possible" requirement means that the error vector is the shortest one possible, it is the unique one that makes a right angle with the prediction vector.(See "The End Goal Diagram" above and notice that the prediction vector is directly beneath the revenue vector, so that the error vector is perpendicular to the prediction vector. If the error vector is perpendicular to the prediction vector it is also *necessarily* perpendicular to both the sales and cost vectors. That is, the error vector is perpendicular to the2-dimensional subspace *plane* swept out by sales and cost vectors.

If two vectors are perpendicular, then their Dot product must be zero.This follows because the Dot product is proportional to the Cosine of the angle between two vectors and perpendicular means 90 degrees, which means Cosine is zero. It works both ways, *if*

vectors are perpendicular then their Dot product is zero and if Dot product is zero then the vectors are perpendicular. That observation is the key to discovering the values of b1 and b2.

■ Perpendicularity is used to Determine b1 and b2

Since, geometrically, the *error* vector is the vector perpendicular to the *prediction* vector (which is expressed by b1 * sales + b2 * cost), as well as being perpendicular to each predictor vector, *sales* and *cost*. I can use that geometric insight which will determine the b1 and b2 that do this.

```
prediction (* here's what I have so far, b1 and b2 are unknown *)
```

```
{-1.22 b1 - 1.2 b2, -0.82 b1 + 0.8 b2, -0.92 b1 - 0.2 b2, 0.48 b1 - 0.2 b2, 2.48 b1 + 0.8 b2}
```

```
error = revenue - prediction (* the error vector must be perpendicular to the
   sales vector and to the cost vector,that fact will determine b1 and b2 *)
```

```
{-2.2 + 1.22 b1 + 1.2 b2, -3.2 + 0.82 b1 - 0.8 b2,
 0.8 + 0.92 b1 + 0.2 b2, 1.8 - 0.48 b1 + 0.2 b2, 2.8 - 2.48 b1 - 0.8 b2}
```

■ Here are the two equations generated by the two Dot Products

```
eq1 = sales . error == 0 (* sales vector is perpendicular to error vector *)
```

```
2.48 (2.8 - 2.48 b1 - 0.8 b2) - 0.82 (-3.2 + 0.82 b1 - 0.8 b2) +
  0.48 (1.8 - 0.48 b1 + 0.2 b2) - 0.92 (0.8 + 0.92 b1 + 0.2 b2) - 1.22 (-2.2 + 1.22 b1 + 1.2 b2) == 0
```

```
eq2 = cost . error == 0 (* cost vector is perpendicular to error vector *)
```

```
0.8 (2.8 - 2.48 b1 - 0.8 b2) + 0.8 (-3.2 + 0.82 b1 - 0.8 b2) -
  0.2 (1.8 - 0.48 b1 + 0.2 b2) - 0.2 (0.8 + 0.92 b1 + 0.2 b2) - 1.2 (-2.2 + 1.22 b1 + 1.2 b2) == 0
```

Now, from these two equations, I need to solve for b1 & b2. I have two equations and two unknowns, so I can do this! I use a built in function of *Mathematica* to solve for the two parameters.

```
Clear[b1, b2] (* clear out any previous calculations for  b1 and b2 *)
```

```
{b1, b2} = {b1, b2} /. Solve[{eq1, eq2}, {b1, b2}][[1]]
```

```
{1.63851, -1.04246}
```

O.k, this says to stretch the sales vector by a multiplier of "1.63851" and reverse and stretch the cost vector by "-1.04246".

```
prediction = b1 * sales + b2 * cost (* Finally!, the prediciton vector *)
```

```
{-0.748021, -2.17755, -1.29893, 0.994976, 3.22952}
```

```
revenue (* repeated here for convenience *)
```

```
{-2.2, -3.2, 0.8, 1.8, 2.8}
```

Correlation Coefficient : Checking out how good the Prediction is

The fairly high correlation, 0.83, shown below, means that the b1 ,b2 combination of the two predictor variables, *sales* and *cost*, do a pretty good job of matching the observed *revenue* vector. All of this is based on the assumption that the relationship is linear. If I can convince myself of that, then the correlation coefficient is highly meaningful. Another common measure of how good the prediction is, is to square the correlation coefficient and get what is called R^2, the coefficient of determination R^2 = 0.68. This means that 68 % of the variability of the revenue is explained by the use of the two predictors sales and cost.

That value, 68 % is not great, but is lots better than just taking and average of the revenue values and calling that the prediction!

```
corrpredictionrevenue = Correlation[prediction, revenue]
 (* correlation coefficient is the cosine between prediction and revenue *)
```

```
0.82878
```

```
R2 = corrpredictionrevenue^2
```

```
0.686876
```

```
angleBetweenPredictionRevenue = ArcCos[corrpredictionrevenue]
  180 / Pi (* angle between prediction vector and the  observation vector *)
```

```
34.0264
```

```
corrpredictioncost = Correlation[prediction, cost]
```

```
0.250719
```

```
angleBetweenpredictioncost = ArcCos[corrpredictioncost] * 180 / Pi
```

```
75.4799
```

```
corrpredictionsales = Correlation[prediction, sales]
```

```
0.941733
```

```
angleBetweenpredictionsales = ArcCos[corrpredictionsales] * 180 / Pi
```

```
19.6554
```

```
corrsalescost = Correlation[sales, cost]
```

```
0.561729
```

```
angleBetweenSalesCost = ArcCos[corrsalescost] * 180 / Pi
```

```
55.8245
```

The important angle here is angle of 34 degrees, between the prediction vector and the observation vector. This is actually just a consequence of the correlation coefficient being 0.828 but still, it's nice to see a physical picture emerge.

Let me collect all the angles as calculated between the various vectors: *revenue*, *sales*, *cost* and *prediction* vectors

.

```
angleBetweenSalesCost
```

```
55.8245
```

```
angleBetweenRevenueSales
```

```
38.6946
```

```
angleBetweenRevenueCost
```

```
78.0071
```

```
angleBetweenPredictionRevenue
```

```
34.0264
```

Drawing the Vector Configurations Using the Correlation Angles - Squashing the Space

Even though the vectors are in 5 - dimensional space, I am going to draw them in standard 3 - dimensional space using the standard X - Y - Z coordinate system. I can do this since the *sales, cost, and prediction* vectors are in a 2-dimensional sub-space of 5-space and so I will place them down in a plane within 3-dimensional space.
In other words, I only need 2-dimensions to represent the sales, cost and prediction vectors so I will use the standard X-Y plane within 3-space to do that. That leaves the third dimension, 'up', the Z axis, available to show the component of the revenue vector not in the plane spanned by the *cost* and *sales* vectors.
. All I am really trying to do is to show the *angles* between the vectors, which only takes 3-dimensions. Notice that I am not dealing with the original *cost, sales, prediction, and revenue* vectors since they are 5-dimensional. What I am doing is to use vectors down in 3-dimensions to *represent* each of the above, *maintaining their relative angles.* .

I know the angle between *sales* and *cost*, which is 55.8 degrees, so I will draw two representative vectors with that spread. I also know the prediction vector is down in the same plane, since it is linear combination of the *sales* and *cost* vectors. It is at 75.5 degrees from the *cost* vector, so I can draw that representative vector down in the same plane as well.

Finally, I know the angle between the *prediction* vector and the *revenue* vector, which is 34 degrees. I will interpret that vector as poking up into the 3rd dimension, the Z-direction, at an angle of 34 degrees. So, my representative revenue vector will be 3-dimensional with its 3rd component poling up at a 34 degree angle.

This configuration is in fact, the basis for the original diagram, "End Goal Diagram", at the start of the tutorial.

A Useful Use of Complex Numbers : Angled Vectors in a Plane (Optional Notes)

I am going to use the properties of complex numbers here just because it's easy to describe unit vectors at various angles in 2-dimensions using complex numbers. For example, the complex number(z = 1 + 1 *I) can be viewed as the vector at 45 degrees of length Sqrt[2], while z = 0 + 1* I is the vector of unit length at 90 degrees. "I" stands for the Sqrt[-1]. Once I have the sales, cost and prediction vectors laid out in two dimensions, I will tack on a zero 3rd component to place them in 3 dimensions. Finally, I know the revenue vector would have the same X-Y components as the prediction vector down in the X-Y plane so that leaves me only how to get its third component up out of the plane. Here's where I use the 34 degree angle between the prediction vector and the revenue vector to determine the amount to poke up into the Z direction.

```
costvec = {1, 0, 0};(* convenient basis vector of  unit length down in the X–Y plane,
representing the cost vector*)
```

```
zsc = Exp[I angleBetweenSalesCost * Pi / 180.]
 (*construct a  sales vector at 55.8245 degrees relative to the cost vector *)
```

```
0.561729 + 0.827321 i
```

Now I' ll construct a sales representative vector with the above X-Y coordinates and a zero Z coordinate.

```
salesvec = {Re[zsc], Im[zsc], 0}
 (*unit vector at 55.8245 degrees  down in the X-Y plane but having a zero Z component*)
```

```
{0.561729, 0.827321, 0}
```

The prediction vector lies in the same plane as cost and sales, so I' ll put it in the X - Y plane as well.

```
zpc = Exp [ I angleBetweenpredictioncost * 180 / Pi]
 (*calc where the prediction vector lays in the X-Y plane as well *)
```

```
-0.276385 + 0.961047 i
```

```
predictionvec = {Re[zpc], Im[zpc], 0}(* unit length *)
```

```
{-0.276385, 0.961047, 0}
```

I know that the revenue vector lies directly above the prediction vector and has the same X-Y coordinates. All I need to do is to find its' height above the X-Y plane . Since I know the angle that the prediction vector makes with the revenue vector, I can use that to compute a height.

```
Tan[angleBetweenPredictionRevenue * Pi / 180.]
```

```
0.67518
```

```
revvec = predictionvec +
  {0, 0,  Tan[angleBetweenPredictionRevenue * Pi / 180.]}
```

```
{-0.276385, 0.961047, 0.67518}
```

Compute Relative Lengths of Cost and Sales Vectors

The relative lengths of two centered vectors can be computed by using their standard deviations. So, based on that, I have made the sales vector some 1.83 times as long as the cost vector. These lengths relate to relative *variability*. These are only relative lengths, but do provide some minor additional insight.

```
stdcost = StandardDeviation [cost] // N
```

```
0.83666
```

```
stdsales = StandardDeviation [sales]
```

```
1.53199
```

```
ratiosalescost = stdsales / stdcost
```

```
1.83108
```

Now draw the vector configurations

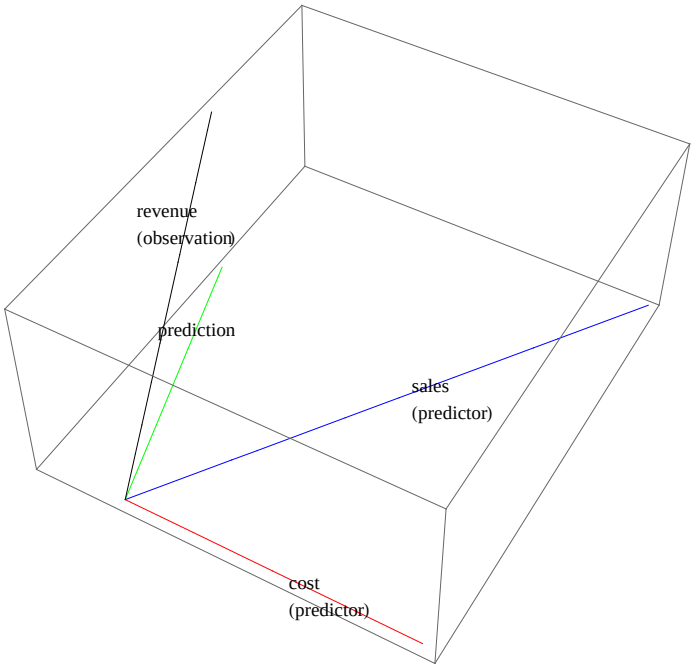
```
predictionRep =
  {Text["prediction", .7 predictionvec ], Green, Line[{{0, 0, 0}, predictionvec}]};
```

```
costRep = {Text["cost
(predictor)", .7 costvec], Red, Line[{{0, 0, 0}, costvec}]};
salesRep = {Text["sales
(predictor)", salesvec + {.1, 0, 0}], Blue, Line[{{ 0, 0, 0},
     ratiosalescost * salesvec}]};
revenueRep = {Text["revenue
(observation)", .7 revvec ], Black, Line[{{0, 0, 0}, revvec}]};

configuration = { costRep, salesRep, predictionRep, revenueRep};

Show[Graphics3D[configuration], PlotLabel →
  "Revenue vector  predicted from Sales and Cost vectors (all data is centered).
Relative lengths (variability) of cost and sales vectors are shown.
 The Cosine of the angle between revenue and prediction vectors is
the multiple correlation coefficient.
   "]
```

Revenue vector predicted from Sales and Cost vectors (all data is centered).
Relative lengths (variability) of cost and sales vectors are shown.
The Cosine of the angle between revenue and prediction vectors is
the multiple correlation coefficient.



Example 2: Predicting Old Faithful's Next Geyser

Old Faithful is a geyser in Yellowstone National Park that periodically erupts in a spectacular fountain of steam and water, delighting everybody nearby. The park rangers post the next eruption time so that tourists can plan to witness it. How can they tell though, when the next eruption will happen? Ahhhh, they use multiple regression!
I will do this prediction in two phases. The first phase uses just one predictor variable to predict when the next geyser will erupt. That 'goodness of fit' prediction will be evaluated using its correlation coefficient. The second phase adds a second predictor variable to see if I can do better.
The raw data set is from Michael Triola's *Elementary Statistics*, tenth edition. I used *Mathematica* to analyze this very small data set plus some geometric insight.... (I checked my geometric result against his algebra and..... they match!)
What I'm looking for is an equation that relates the time until *next* eruption to the *duration* and *height* of the current eruption.

Here is the raw data from Triolas book pg 515, I'm only going to use three columns of data: time to next (eruption), duration, and height data. (Notice that I take each column of data and interpret it as a vector).
Since I am going to transform this data later, I will describe it as being the 'raw' data or the raw vectors. So, the *rawnext* vector has 8 components or elements. The first element is the number 92. The corresponding first element of the *rawduration* vector is the number 240 and the corresponding first element of the *rawheight* vector is 140.
That means that there was a 92 minute interval from a previous eruption that was characterized by a duration of 240 seconds and reached a height of 140 feet. Spectacular! The last element of the *rawnext* vector is 87 which means there was an 87 minute interval from the current eruption that lasted 220 seconds, and reached a height of 150 feet.

```
rawnext = {92, 65, 72, 94, 83, 94, 101, 87};
(*  data vector that showed time until next/following  eruption, in minutes -- *)

rawduration = {240, 120, 178, 234, 235, 269, 255, 220};
(* duration vector  of  duration of current eruption, (how long it lasted)  in seconds *)

rawheight = {140, 110, 125, 120, 140, 120, 125, 150};(* vector  of
 height of current eruption in feet *)
```

- **Presenting the full raw data in tabular format**

```
labels = {"next(min)", "duration(sec)", "height(ft)"};

rawdata = Transpose[{rawnext, rawduration, rawheight}]
```

```
{{92, 240, 140}, {65, 120, 110}, {72, 178, 125}, {94, 234, 120},
 {83, 235, 140}, {94, 269, 120}, {101, 255, 125}, {87, 220, 150}}
```

```
Join[{{"    ", "Raw Geyser Data", " "}}, {labels}, rawdata] // Grid
```

	Raw Geyser Data	
next(min)	duration(sec)	height(ft)
92	240	140
65	120	110
72	178	125
94	234	120
83	235	140
94	269	120
101	255	125
87	220	150

Looking at the raw data above, the first row says that there was an eruption of 240 seconds reaching a height of 140 feet. The time

until the following/next eruption was 92 minutes. The last row says that an eruption lasted 220 seconds, reached a height of 150 feet, and the next eruption was 87 minutes later.

Phase I: Using One Predictor variable, "Duration" to Predice "Next"

I am going to step though the process for determining the connection between two variables, *next* and *duration*. This is an example of what is called Bivariate Regression. I am going to predict the time until the *next* eruption based on the current *duration* of an eruption. This result will give me a line, a 'trend' line that will tell me: knowing *duration* will give me time to *next* eruption. That connection between *next* and its predictor variable, *duration*, can be expressed as a straight line equation as below : Plugging in any know or conjectured *duration*, will tell me the *value* of *next* (if , of course, I know the coefficient b1). Further, the Cosine of the angle between the *next* vector and the prediction vector, which will be b1 * *duration*, is the correlation coefficient and will tell me how good the prediction is.

Center the Data Before Any Further Calculations, since the data must be centered for the geometric interpretation to be valid.

```
next = rawnext - Mean[rawnext] // N;(*centered data - time until next eruption (min)*)

duration = rawduration - Mean[rawduration] // N;(*length of eruption (sec.) *)

height = rawheight - Mean[rawheight] // N;(* not= { next,  used in Phase I *)

centeredbivariatedata = Transpose[{next, duration, height}];

labels = {"next", "duration"};

Join[{{"    ", "Centered Geyser Data", " " }}, {labels}, centeredbivariatedata] // Grid
```

Centered Geyser Data		
next	duration	
6.	21.125	11.25
-21.	-98.875	-18.75
-14.	-40.875	-3.75
8.	15.125	-8.75
-3.	16.125	11.25
8.	50.125	-8.75
15.	36.125	-3.75
1.	1.125	21.25

The plan is to calculate a *prediction* vector (which is simply a multiple of the *duration* vector) that will be as close as possible to the *next* vector. The equation I am looking for will be as below:

prediction = b1 * duration (* b1 is a constant to be estimated from the data columns*)

Step 2 : The vectors here are 8 component vectors which puts them in 8-dimensional space, but, since there are only two vectors here, I can draw them in 2 dimensional space . This is so since these two vectors form a 2-dimensional *sub-space* within an 8-dimensional space!
Drawing these two vectors and looking at their relation to each other, gives me additional insight complementary to what I can get from a statistics package. The primary relation I want to look at here is the angle between them. If the angle is small, then the *duration* vector is a good predictor of the *next* vector. If the angle is large, then knowing the *duration* vector doesn't help much to know the *next* vector. In fact, if the angle between these two vectors is 90 degrees, knowing *duration* doesn't help at all, and you are looking at *independence between these variables.*

Step 3: Calculate the angle between the *next* and *duration* vectors
The cosine of the angle between these two vectors is called the correlation coefficient (usually denoted as 'r') which we will

visualize presently. The square of 'r' is .... called "R^2" and is the proportion of the variability of the *next* vector that is explained by the *duration* vector.
From the calculations below you can see that the angle between the next vector and the duration vector is about 22 degrees, which is quite good. *Maybe that's good enough and I don't need to add on additional predictors!*

```
corrnextduration = Correlation[next, duration]
```

```
0.925591
```

```
theta = ArcCos[corrnextduration] * 180 / Pi(* degrees between duration and next *)
```

```
22.2423
```

The bivariate regression equation, using centered variables

We are going to estimate the time interval before the next eruption by using values of duration of eruption. We don't know b1, but will estimate that from the geyser records.

```
Clear[b1]
```

```
prediction = b1 * duration (* this is an extension of the
   duration vector that puts its tip directly under the next vector. *)
```

```
{21.125 b1, -98.875 b1, -40.875 b1, 15.125 b1, 16.125 b1, 50.125 b1, 36.125 b1, 1.125 b1}
```

```
error = next - prediction
(*this expresses the minimum distance
 from the tip of nextEstimate up to the next vector *)
```

```
{6. - 21.125 b1, -21. + 98.875 b1, -14. + 40.875 b1, 8. - 15.125 b1,
 -3. - 16.125 b1, 8. - 50.125 b1, 15. - 36.125 b1, 1. - 1.125 b1}
```

The direction of the duration vector is perpendicular to the direction of the error vector and so its' dot product will be zero. This results in an equation for b1, which I can solve.

```
eq1 = duration . error == 0
```

```
50.125 (8. - 50.125 b1) + 36.125 (15. - 36.125 b1) +
  21.125 (6. - 21.125 b1) + 16.125 (-3. - 16.125 b1) + 15.125 (8. - 15.125 b1) +
  1.125 (1. - 1.125 b1) - 40.875 (-14. + 40.875 b1) - 98.875 (-21. + 98.875 b1) == 0
```

The regression coefficient

```
b1 = b1 /. Solve[eq1, b1][[1]] (* solve for the b1 constant that
   minimizes the distance from the nextEstimate up to the  next vector *)
```

```
0.234061
```

Now, the prediction vector is:

```
prediction
```

```
{4.94455, -23.1428, -9.56726, 3.54018, 3.77424, 11.7323, 8.45547, 0.263319}
```

and the prediction equation is :

```
nextvalue = b1 * durationvalue
```

```
0.234061 durationvalue
```

Visualizing the Correlation Coefficient

Let me plot the two centered vectors *next* and *duration*. What should they look like? Remember, they could have many components and, in fact, these particular vectors have 8 elements. How to plot those? Since these two vectors sweep out a plane in 8-dimensional space, it is really necessary only to plot them in that plane! So, the picture can be relegated back to plain old 2-dimensional space.

When I plot them I will use their standard deviations as a measure of their relative lengths. I calculate these below. Since the standard deviation of the *next* vector is about 12 while the standard deviation of the *duration* vector is about 48, Their ratio is about 4 to 1.

```
stdNext = StandardDeviation[next]
```

```
12.1655
```

```
stdDuration = StandardDeviation[duration]
```

```
48.1083
```

```
ratioDurationNext = stdDuration / stdNext
```

```
3.95448
```

Ok, now I want to draw the next vector and the prediction vector. I don' t need to use 8 - dimensional space since these two vector can be considered to lie within a 2-dimensional X-Y subspace of the 8-dimensional space. Further, I don't need their actual lengths, since all I want the relative angles between them. A small additional bit of infiormation is that their lengths can be drawn as a ratio of their variabilities.

To keep the prediction vector distinguishable from the X axis, I have off set it just a bit.

```
offsetAngle = 10 (* 10 degreees, this will get  the prediction vector off the  X-axis *)
```

```
10
```

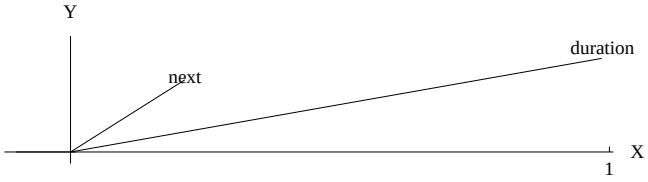
```
z1 = Exp[ I offsetAngle * Pi / 180. ] (*next pseudo vector *)
```

```
0.984808 + 0.173648 i
```

```
z2 = Exp[I (theta + offsetAngle) * Pi / 180.](* prediction pseudo vector ,
 the vector that is theta degrees from the prediction vector*)
```

```
0.845799 + 0.533501 i
```

```
Graphics[
 {Line[{{-.10, 0}, {0, 0}}],
  Line[{{0, 0}, {Re[z1], Im[z1]}}], Text["duration", {Re[z1], Im[z1] + 0.02}],
  Line[0.25 * {{0, 0}, {Re[z2], Im[z2]}}], Text["next", .25 * {Re[z2], Im[z2] + 0.02}]
   }
 , Axes → True, AxesLabel → {"X", "Y"}
 , Ticks → {{1, 2, 3, 4}, {0, .25, .5}}, PlotRange → All]
```



Writing the Raw Data Prediction Equation

If I wanted to express the prediction equation in terms of the raw data, I need to calculate a constant, b0. Fortunately, that is very easy as shown below. I calculated b1 from centered data and now I use that to back into the original raw data equation .

Relating the Regression Equation Back to the Original Raw Data

Up to this point I have used centered data and calculated b1. You might want to know how to express the regression equation using the original data. That turns out to be super easy and only requires that you calculate one more constant 'b0'. That is an easy calculation shown below.

```
b0 = Mean[rawnext] - b1 * Mean[rawduration]
```

```
34.7698
```

```
rawPrediction = b0 + b1 rawdurationVariable (* this is the raw data regression line*)
```

```
34.7698 + 0.234061 rawdurationVariable
```

Phase II: Using Two Predictors, Duration and Height

Now I' m going to use two predictor variables, the *duration*, as well as the *height* of the water plume. I suppose you could call this a *trivariate regression*, but I won't. The usual description would be to call this a *multivariate regression*. The hope is to get an even tighter match between the *next* value and some combination of *duration* and *height*.

Step 2: Now calc the angle between each vector above by using the correlation coefficient
That is, if I know the correlation coefficient between each vector, I can then calc the angle (in degrees) between them.

```
angledurationheight =
 180 / Pi * ArcCos[Correlation[duration, height]] (*begrees between duration and height *)
```

```
67.0802
```

```
angledurationnext =
 180 / Pi ArcCos[Correlation[duration, next ]](* degrees between duration and next *)
```

```
22.2423
```

```
angleheightnext = 180 / Pi ArcCos[Correlation[height, next ]](* degrees between
   height and next vectors *)
```

```
74.3733
```

Note that these three angles pin down the directions of the three vectors relative to each other. The geometry will tell you when certain combinations of angles are possible or not (Venn diagrams can't do this).

Now to actually get b1 and b2, I make use of the fact that the 'next'
vector minus it's estimate, 'prediction' is the 'error' vector. That error vector is perpendicular to the plane generated by the duration and height vectors and is therefore perpendicular to each.
Taking each dot product and setting that to zero
gives me two equations in the two unknowns b1 and b2. Voila! We're (almost) done.

```
Clear[ b1, b2, prediction] (*clear out any previous values *)
```

```
prediction = b1 duration + b2 height;

error = next - prediction;

eq1 = height . error == 0;

eq2 = duration . error == 0;
```

This gives me two equations and two unknowns, b1 and b2.

```
soln = Solve[{eq1, eq2}, {b1, b2}][[1]]
```

```
{b1 → 0.244636, b2 → -0.0982503}
```

```
{b1, b2} = {b1, b2} /. soln
```

```
{0.244636, -0.0982503}
```

' prediction' is the vector, down in the plane spanned by *duration* and *height* vectors,
that is closest to the ' next ' vector. That is, the tip of the *prediction* vector is directly under the *next* vector. The distance between them is the 'error' made in the estimate.

```
prediction = b1 durationVariable + b2 heightVariable /. soln
```

```
0.244636 durationVariable - 0.0982503 heightVariable
```

However, this is the centered multi-regression eqn, now I need to add back in the constant term to get the uncentered equation.

```
b0 = Mean[rawnext] - (b1 Mean[rawduration] + b2 Mean[rawheight])
```

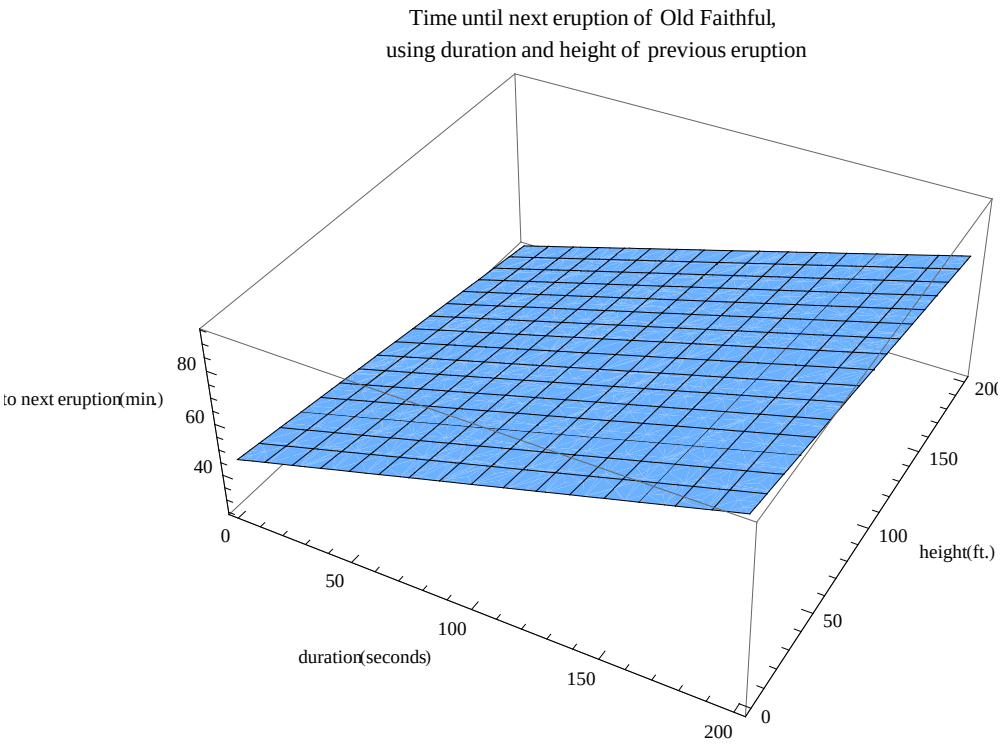
```
45.1049
```

Time to next eruption -- The equation we have is a 'plane' a 2-dimensional figure embedded within 3-dimensional space.

```
nextEruptionTime = b0 + prediction
```

```
34.7698 + 0.244636 durationVariable - 0.0982503 heightVariable
```

```
Plot3D[nextEruptionTime, {durationVariable, 0, 200}, {heightVariable, 0, 200},
 PlotLabel → "Time until next eruption of Old Faithful,
using duration and height of previous eruption",
 AxesLabel → {"duration(seconds)", "height(ft.)", "time to next eruption(min.)"}]
```



Example: What is the time to the next eruption? If I plug in a duration of 180 seconds
for the last eruption that had a height of 130 ft.
then this says that the next eruption will be 76 minutes from that last one.

```
a + nextEstimate /. {durationVariable → 180, heightVariable → 130}
```

```
76.3669
```

How Good is our Estimate? Ok, I have an estimate of the next time until an eruption. How good does that estimate seem to be? The correlation coefficient give a partial answer. This is the cosine of the angle between the 'next' vector and the 'nextEstimate' vector. Remember the nextEstimate vector is directly under the 'next' vector. There is an angle,
call it 'theta', between them can be gotten from the cosine of the angle between them. That cosine is called the Multiple Correlation Coefficient and I calc it below.

```
multipleCorrelationCoefficient =
 next . (b1 duration + b2 height) / (Norm[next] Norm[b1 duration + b2 height])
```

```
0.93086
```

```
theta = 180 / Pi * ArcCos[multipleCorrelationCoefficient ]
```

```
21.4307
```

So, the angle between the 'next' vector and its estimator,
nextEstimate is about 21 degrees, pretty close! Look back at the angles between duration and next. It was about 22 degrees.

Hmmmmm, looks like the duration accounts for a lot of the variability in the time for next eruption. Maybe just a simple bivariate regression of next versus duration would work just as well for us?

Summary of Old Faithful

So, the analysis consists of finding b1, b1, and b2 so that all of the values on the left hand side are matched as closely as possible. The left hand side is a vector, and the right hand side is too. There are several ways to find b0, b1, and b2 with the most popular know as least squares analysis. You can calculate these parameters by hand, as I will show later using geometry, or you can plug these numbers into a stat package, push a button, and get lots of valuable information that you will then need to interpret.

Phase 2 : Looks at that equation above and says - sure you can write down anything, but how accurately does that equation match what actually happened, that is how closely does it *correlate*? That is, given the vector of observed Revenues, how close does the linear combination of the other two vectors match this? This phase requires a judgement as to how closely the observation vector is correlated with the combination of the predictors.

The error in prediction relates to the distance between the tips of these two vectors. Or, another way to see this is to look at the angle between these two vectors, the smaller the angle, the better the prediction. The cosine of the angle between these two vectors is called the correlation coefficient, a most important concept in statistics.
The take home idea here is that this correlation is a simple geometric judgement involving finding the angle between the observation vector and the vector that is the combination of the predictor variables.

References

Wickens, Thomas (1995), *The Geometry of Multivariate Statistics*, Erlbaum Publishers
Rucker, Rob (2007) Trigonometry Basics, on milagrosoft website
Rucker, Rob (2008) Vector Arithmetic and Vector Operations, on milagrosoft website.
Rucker, Rob (2008) Business Trend Analysis, on milagrosoft web site.