Long Live (Big Data-fied) Statistics!

Norman Matloff[*]

Abstract
Big Data is of intense interest today, in business, government, health care and so on. Though many of the problems are inherently statistical, the center of gravity of research in the field has moved to computer science. Indeed, some claim the topic to be outside the realm of statistics entirely. Yet it is shown here that Big Data makes statistics more important than ever. Statistical methods, and even more importantly, statistical concepts, are vital to effective use of Big Data. Equally, parallel processing methods are needed to make it all work. This paper takes a statistical approach to two major problems with Big Data: visualization and the Curse of Dimensionality.

Key Words: Big Data, visualization, nonparametric curve estimation, Curse of Dimensionality, parallel processing

1. Introduction

Recently a friend who works in "analytics," ferreting out relations in highly voluminous market data, mentioned to me, "With Big Data one no longer needs statistics." He was referring to statistical inference—confidence intervals and significance tests—becoming unnecessary since the large sample sizes may give standard errors that are effectively zero.

On another occasion, I was told a young analyst had gushed, "With machine learning techniques, one doesn't need statistics," apparently using the latter term to mean classical parametric methods.

Meanwhile, Davidian (2013) lamented the fact that the statistics profession had somehow let its rightful claim to Big Data slip away, grabbed by other fields, again notably computer science.

To be sure, as of this writing (summer 2013) some statisticians are reclaiming the field. JSM 2013 was full of papers containing "Big Data" in their titles. But more needs to be done, especially at the core—demonstrating that many Big Data issues are, yes indeed, statistical in nature, and that understanding that nature is key to making effective use of large data sets.

To begin with, the "zero standard error" argument fails when one considers the fact that with Big Data, it is often the case that the number of variables is a large proportion of, and even much greater than, the number of observations. This does not lead to near-0 standard errors.

And as to machine learning methods obviating the need for statistics, it's unfortunate that so many ML users don't realize that they *are* doing statistics. Most ML methods are variants of nonparametric methods, such as k-Nearest Neighbors, that have been part of statistics for decades.

It will be assumed throughout that the data has the familiar (though by no means universal) "n by p" structure, i.e. n observations on p variables. This paper's contributions consist of the following:

- It will be argued that of the two Big Data problems, Large n and Large p, the former is fairly easily handled, while the latter will continue to be a major challenge.

[*]Dept. of Computer Science, University of California, Davis

- New visualization tools will be presented, aimed at Large n settings, and making a dent in the Large p problem.
- A novel approach to the Curse of Dimensionality problem will be proposed.

2. Large-n Problems

The old days of textbook problems having n = 25, with n often not being much larger in real life, are long gone. Values of n in the tens of thousands are quite common, and some data sets have n in the hundreds of millions or even larger.

There are two major problems with Large n, one obvious and the other perhaps less so:

(a) Large-n settings can require very large amounts of computation.

(b) Traditional visualization techniques in Large-n settings tend to suffer from the "black screen" problem, in which so many data points are plotted that all or part of the screen becomes solid black, losing most of its value.

Both of these problems are manageable, as we will now see.

2.1 Large-n: Computation

The Large-n case is certainly raises computational issues, especially in light of the fact that much analysis is now done in distributed data settings such as Hadoop. Any computation that is not *embarrassingly parallel*–i.e. easily broken up into independent chunks whose computation does not involve much interaction between the chunks—can be prohibitively time-consuming. However, this is less a problem than one might guess.

First, if standard errors are indeed known to be small (n large, $p << n$), simply applying the estimator to a single random subsample of the data may suffice.

But even outside of that setting, note that many classical procedures, such as linear models, general linear models, principle components analysis and so on, rely only on sums and sums of squares, whose computation is embarrassingly parallel.

Moreover, most statistical procedures that assume i.i.d. data can be replaced by an embarrassingly parallel method that has the same statistical accuracy as the original one, as described in general in (Matloff, 2013), and with similar results for the linear regression case in (Fan *et al* 2007) and (Guha, *et al* 2009). The idea is simple: Break the data into chunks, apply the given estimator to each chunk, then average the results. Chunking is of course a standard strategy for parallelizing algorithms, but the key point here is that (Matloff, 2013) shows that this method yields the same asymptotic standard errors as produced by the original estimator.

In this way, non-embarrassingly parallel problems are converted to parallel ones, which I call Software Alchemy. Here is an example of speedup, from (Matloff, 2013):

Hazard function estimation, regression (n = 50000, 0.20 quantile):

cores	speedup	rel diff
2	1.87	0.0017
4	3.25	0.0055
8	6.86	0.0074
16	9.02	0.0100
24	9.68	0.0091
32	11.69	0.0079
40	10.97	0.0108

So even on a quadcore machine, we get a speedup of more than 3, and the average relative difference between the original estimator and the converted one is only .0055.

2.2 Large-n: The Black-Screen Problem

Consider for example the classical scatter plot of variables X and Y. With even a moderately large n, the plot will be so full of dots that portions will be entirely solid black, thus essentially meaningless. Note that this can occur even for moderately large values of n, due to the nonzero width of a pixel.

The solution is to actually *exploit* the Large-n property. A scatter plot essentially is estimating the (two-dimensional) density $f_{X,Y}$. Thus we might as well explicitly recognize that fact, and construct a density estimate in the first place.

We could do this parametrically, say by assuming a bivariate normal distribution, or possibly a copula model. Or, we could use a completely nonparametric estimation method, such as the standard kernel or nearest-neighbor techniques. With Large n, nonparametric density estimation methods are feasible.

The function **scatterSmooth;()** in the R language takes the nonparametric approach. This notion recently been extended in the R **bigviz** package, (Wickham, 2013). This approach is also used in the novel visualization methods proposed here in Section 3.

One might at first guess that we need three dimensions to plot the density estimate, with the third one being the height of the density. But this problem is solved by color-coding that height, essentially the familiar heat map.

3. Large-n or Large-p: Existing Visualization Methods

A central problem with visualization in Large-p settings is that our displays are two-dimensional, thus set up only for the case p = 2. Let's look at two (of many) methods that have been developed to circumvent this problem.

3.1 Two Existing Methods

- *Parallel coordinates (Unwin et al, 2006):* Here we draw p vertical axes, one for each variable, For each of our n data points, we draw a polygonal line from each axis to the next. Say for instance p = 3, and we have data on people's height, weight and age, and that one person in our data has height 70 inches, weight 150 pounds, and age 32. We would draw a dot at 70 on the first axis, at 150 on the second, and at 32 on the third, and then draw a polygonal line connecting the three dots. The R function **parcoord()** in the MASS library plots this, for instance.

  The parallel coordinates approach can be very revealing if both n and p are small. But for Large n, the black-screen problem hits us with a vengeance, as seen in Figure 1 for the baseball data from the UCLA Statistics Department,[1] with n being only 1033. Moreover, for even moderate values of p, relations between variables whose axes are not near each other are very hard to discern.

- *Grand tours (Wegman, 1992):* This method consists of repeated rotations and projections of the p-dimensional data into the two-dimensional screen. R's **tourr** library implements this. Again we have the black-screen problem, and interpretation is difficult.

[1] `http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights`

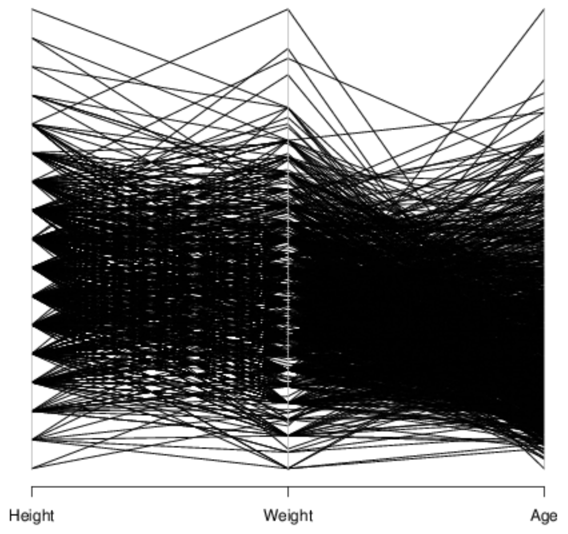

Figure 1: Baseball data: height, weight, age

The black-screen problem is not hopeless, and methods have been proposed for dealing with it in the case of parallel coordinates (Unwin *et al* 2006), but it is still has been a tough problem. In Section 3.3, I propose a novel alternative approach.

That approach and other methods are included in my new R library, **BigNGraphs**.

3.2 Boundary Curves As Descriptive Tools

Suppose we are exploring the relationship between variables X, Y and Z. Consider the *boundary curve*

$$\{(u,v) : E(Z|X = v, Y = v) = b\} \tag{1}$$

for some user-specified b. This defines a curve in the u-v plane.

For the moment, let's suppose Z is an indicator variable, 1 for membership in a certain class and 0 if not, so that (1) reduces to the conditional class probability,

$$\{(u,v) : P(Z = 1|X = v, Y = v) = b\} \tag{2}$$

Then we might take $b = 0.5$, for instance, which corresponds to the familiar classification problem: Say have a new observation with X and Y known but Z unknown and to be predicted. If (X,Y) is on one side of the curve, we predict Z = 1 while on the other side we predict Z = 0. Many statistical models—logistic regression, for instance, and also SVM after a pre-transformation—assume that the boundary curve has a straight-line form, but in the general case (i.e. not assuming any particular form for the curve) one can estimate the curve using nonparametric methods.

In addition to prediction, we can use the boundary curve for understanding, i.e. description. With the logistic model, for example, the estimated coefficients are often used to describe how X and Y affect Z. If we estimate the curve nonparametrically, then entire curve is used for description, as in (Hastie *et al*, 2001). This paper extends that work, by proposing the simultaenous graphing of *several* boundary curves as a means of circumventing the two- dimensional screen problem, as will be seen below.

Another possible choice for b is the estimated value of P(Z = 1), the unconditional probability of being in the given class. Points on one side of the boundary curve are now those for which the class probability is higher than average, with points on the other side having a lower-than-average value.

For general Z (i.e. not necessarily an indicator random variable), we might set b to be our estimated E(Z), the overall unconditional mean of Z. Points on one side of the boundary curve are now those for which the conditional mean is higher than average, with points on the other side having a lower-than-average value.

3.2.1 Estimation

My software, available at `http://heather.cs.ucdavis.edu/bigngraphs.html`, takes the following approach to estimating a boundary curve:

- For each point in our data set, a nonparametric estimate of the regression function of Z given X and Y is calculated at that point.
- It is determined which points have an estimated regression value near b, thus finding a band around the estimated boundary.
- The points in the band are smoothed, producing the estimated boundary curve.

3.2.2 Dimensional Impacts

Plotting a boundary curve allows us to plot three variables on a two-dimensional display. Actually, we could also do this by plotting the regression function of Z on X and Y itself, representing the regression surface height by color coding. That would be nice (and there is a function to do this in my software suite), but it would not permit intergroup comparison.

The key to the proposed method here is that instead of plotting an entire surface, we plot a boundary curve. This enables us to display the effects of more variables, in the form of groups.

Consider for instance a fourth variable, G, representing group membership, say race, political party affiliation or whatever. If there are, for example, three groups, we might plot three (X,Y,Z) boundary curves, one for each group. In this way, we are plotting four variables on a two-dimensional screen. If we were to break things down further by introducing another group G_2 (renaming G to G_1), we would now be plotting five variables on a two-dimensional screen.

An alternative would be to plot estimated regression curves—Z against X and Z against Y—within each group or group combination. With two group variables, G_1 and G_2, we would still be plotting five variables in two dimensions. But we would lose the interaction between X and Y in their effects on Z. In other words, the proposed method should give us richer insight into the data.

3.2.3 Example: Bank Data

This example involves the Bank dataset from the UC Irvine data repository.[2] A bank developed a new product, a special kind of account, and marketed it to its existing customers. The goal of the data is to determine which factors affect a customer's decision to sign up for

[2]`http://archive.ics.uci.edu/ml/datasets/Bank+Marketing`

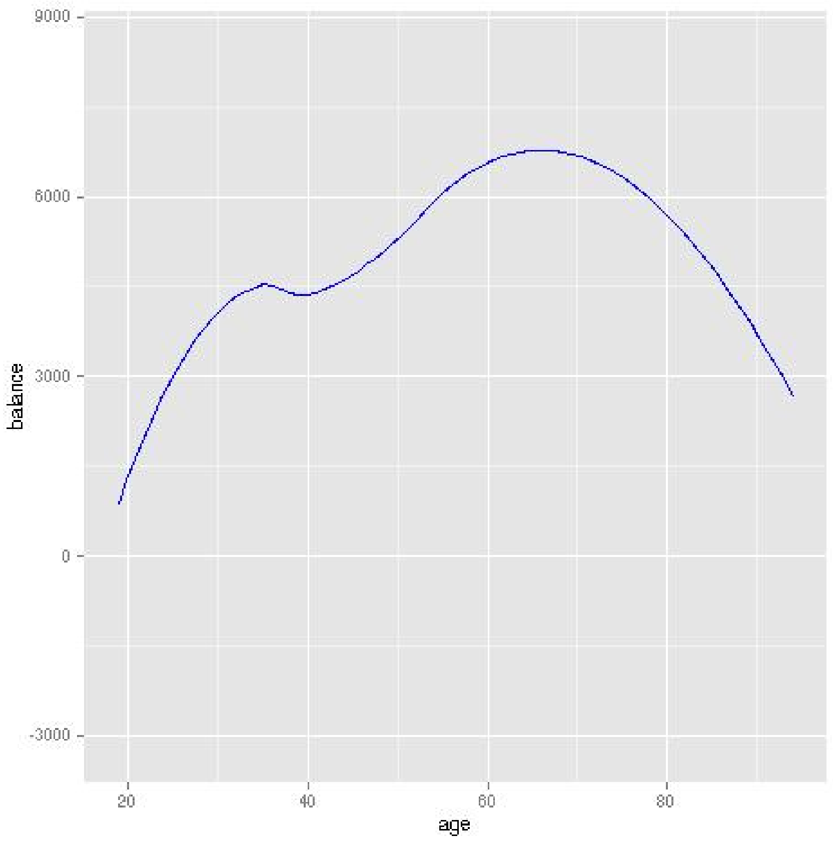

Figure 2: Bank data

the new type of account. In the analyis here, X is age of the customer, Y is the customer's current balance, and Z is in indicator random variable representing acquisition of the new account. The boundary curve is shown in Figure 2.

Keep in mind the meaning of the curve: Customers who are above the curve—i.e. have (age, balance) combinations above the curve—have a higher than average probability of agreeing to switch to the new account. Customers below the curve are less likely to switch.

A salient feature of the graph is that it peaks around age 60, near retirement age. Customers of this age are seen to be "harder sell": They need a higher balance than others before agreeing to switch. It could be that the more favorable terms of the new account would not make much difference to these customers, as they intend to close their accounts upon retirement. In any case, this graph should be brought to the attention of domain experts.

3.2.4 Example: Income Data

The real value of the boundary method is in comparing several groups, as seen in this example.

Here we have another UCI data set, known as Adult.[3] The topic of interest is the factors affecting the probability of having a high income. X and Y were age and years of education, respectively, Z was an indicator variable for high income, and G_1 was a group variable for gender. So here we are plotting four variables in two dimensions, as seen in Figure 3.

[3]`http://archive.ics.uci.edu/ml/datasets/Adult`

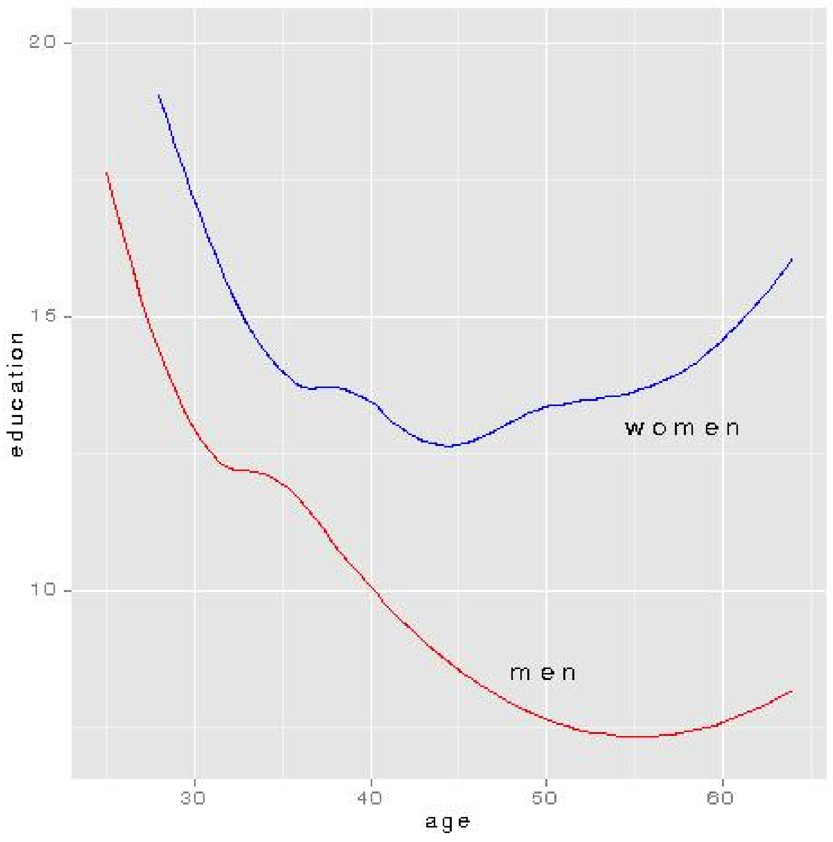

Figure 3: Adult data

The results are troubling. We have all heard about the gender gap in pay, but the graphs here reveal that the gap is not uniform with respect to age. There is not much difference before age 35, but after that the difference is dramatic; in order to earn high income, women of a given age need much more education than men.

3.2.5 Example: Airport Data

This involves the flight lateness data, `http://stat-computing.org/dataexpo/2009/`. I used just the year 2008, and ran boundary curves for the airports Dulles, Houston and San Francisco. Z, X and Y were arrival delay, departure delay and flight distance, respectively.

Here, a boundary curve represents the various (DepDelay,Distance) combinations that yield a mean value of ArrDelay of 10.0. A higher curve is better, as it means that, for a given value of departure delay, one needs a longer flight distance before having an expected arrival delay of more than 10 minutes.

The results, shown in Figure 4, suggest that Dulles (blue) is faring less well, while San Francisco (green) is doing the best.

3.2.6 Parallel Computation

I chose k-NN for the nonparametric regression procedure, in order to use a Fast Nearest Neighbor (FNN) algorithm (Li, 2013) as its core. This brings some improvement in speed, but given that we are working with Big Data, with potentially very large computing needs,

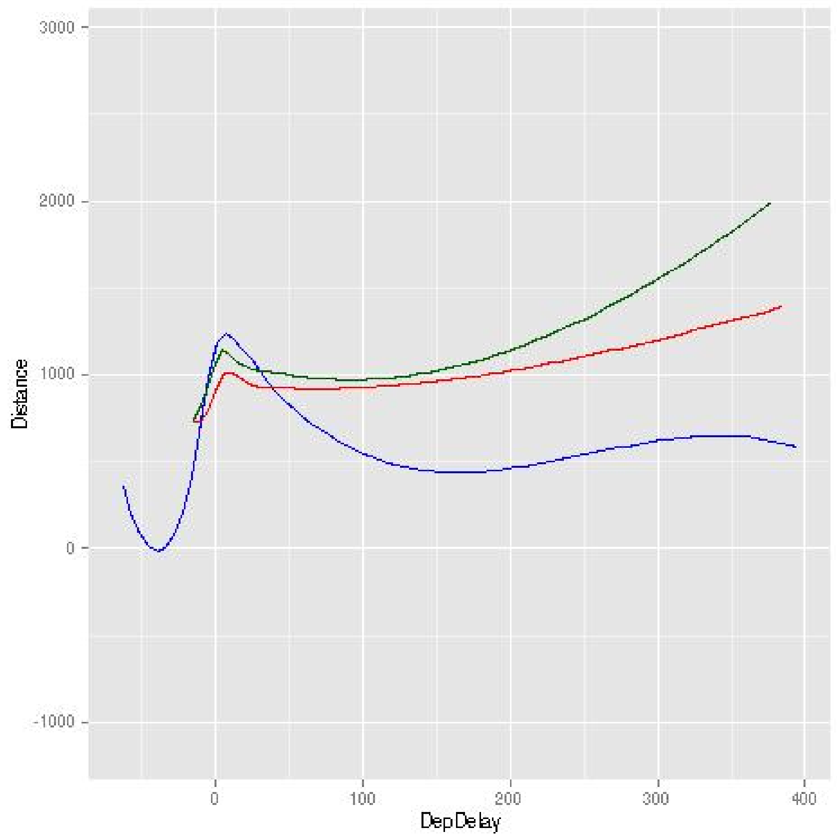

Figure 4: Airport data

I use parallel processing for the computation. The current mechanism for parallelization is the library named **parallel** in base R, which allows for either shared memory or distributed computing.

A number of parallel algorithms have been developed for nearest-neighbor computation, such as (Garcia *et al*, 2008). But I take a data-parallel approach, both for conveience and to be able to leverage distributed databases, such as Hadoop, if we are on such systems.

Specifically, the chunking method described in Section 2.1 is used. In this setting, this means doing the k-NN computation only *within chunks*, not across them. This produces acceptable accuracy while greatly reducing computation.[4]

3.3 A Novel Approach to Parallel Coordinates

My theme so far has been to apply nonparametric curve estimation methods to deal with the black-screen problem. I now apply this to parallel coordinates.

A number of authors have applied nonparametric density estimation methods to the parallel coordinates problem; see (Artero *et al*, 2004), for a list of references. However, my approach is very different, addressing that problem by plotting only a very small number of"typical" lines, defined as those that have the highest estimated multivariate density.

This approach, implemented in the function **freqparcoord()** in my **BigNGraphs** pacakge, can be very advantageous for many data sets, as it addresses both major drawbacks of parallel coordinates:

[4]See (Matloff, 2013) for the theoretical underpinnings of this approach. Note that in the term *chunk averaging*, the averaging is in essence occurring later, during the smoothing of the boundary band.

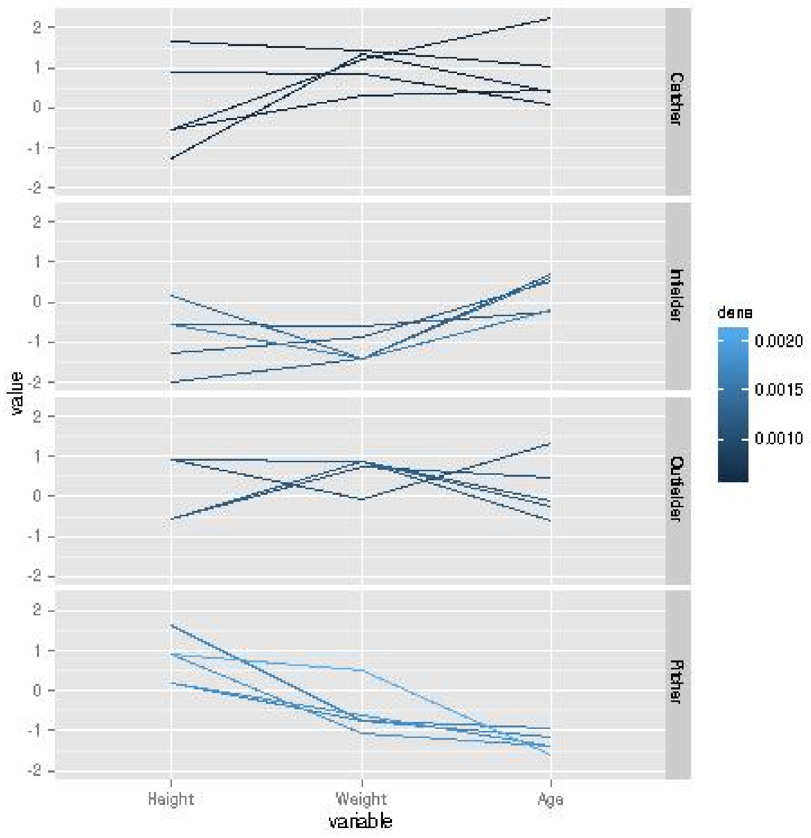

Figure 5: Baseball data under new approach: height, weight, age

- The black-screen problem is completely avoided, since we plot only a small number of lines.
- The fact that only a few typical lines are plotted makes it easier to notice relationships between far-apart variables, the other major drawback of the classical parallel coordinates method.

The function also allows graphing a random subset of the data, with lines color-coded by density estimate value.

3.3.1 Example: Baseball Player Data

This example uses the aforementioned baseball player data set, included in the **BDGraphs** package, courtesy of the UCLA Statistics Dept. Figure 5 shows the results under the new approach, graphed by type of player position.

Remember, the plotted lines are "typical," i.e. they are the ones having the maximal density in each group. We see a marked contrast between the various positions. Pitchers are typically tall, relatively thin, and young. By contrast, catchers typically are much heavier, and substantially older; their typical heights have more of a range. On the other hand, compared to catchers, infielders are typically shorter, lighter and younger.

Plotting the full data by player position is much less revealing, even if density is used to plot, which in effect is accomplished via alpha blending (Unwin *et al*, 2008). See Figure 6, where alpha = 0.2. This view seems much less revealing than the above approach of plotting "typical" lines.

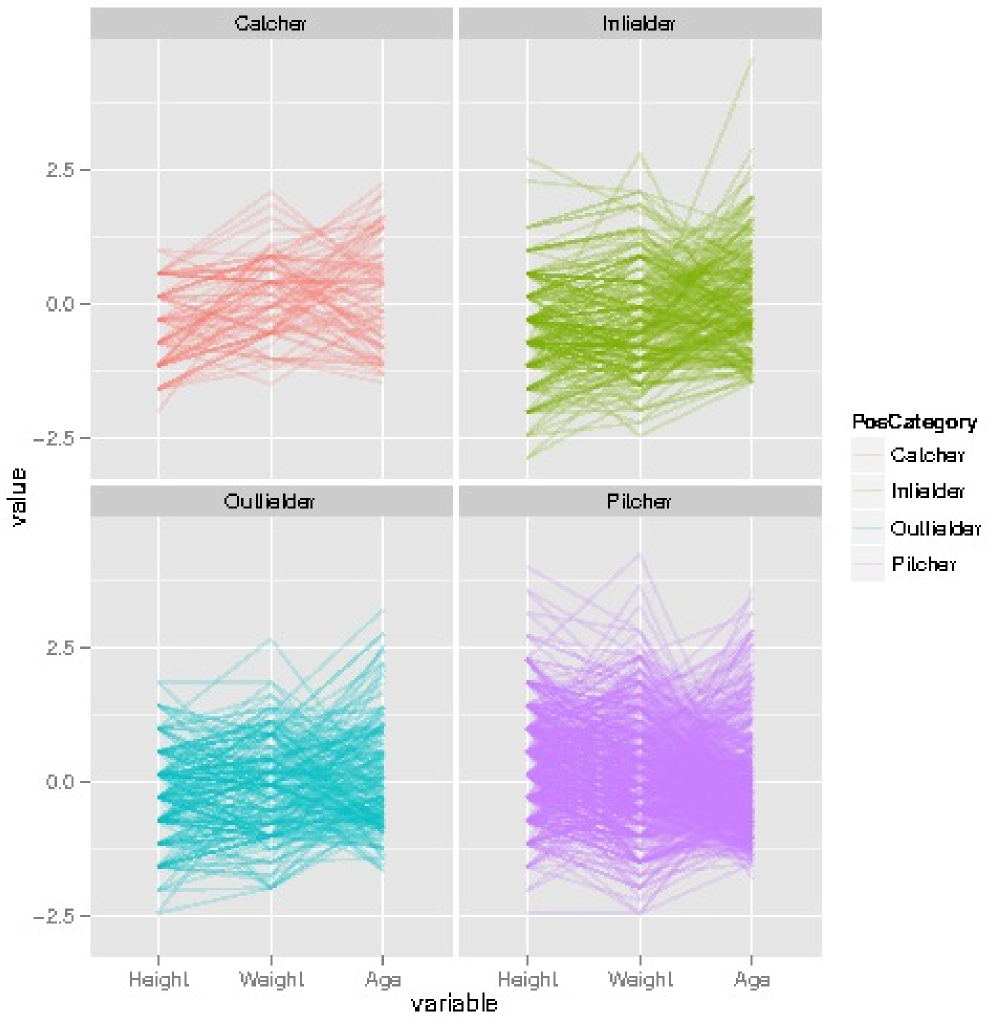

Figure 6: Baseball data, alpha = 0.2

3.3.2 Example: Letter Recognition Data

The variables from this UC Irvine site concern various properties of images of letters, in an attempt to doing machine recognition of them.[5] See the UCI site for details. Figure 7 shows the **BigNGraphs** parallel coordinates plot.

Analysis of these lines is really for a domain expert, yet it is interesting to see how similar the typical lines are for the letters 'O' and 'Q'—two letters that differ only by one stroke—yet there is substantial divergence between two other letters that differ by a single stroke, 'E' and 'F'.

4. Large p and the Curse of Dimensionality

The dreaded Curse of Dimensionality (COD) is famous among those who work with Big Data. Yet truly understanding it, and thus dealing with it effectively, once again depend on a solid statistical understanding of the problem, and indeed careful, thoughtful consideration.

For example, an oft-cited manifestation of the COD is that the volume ratio of the unit

[5]`http://archive.ics.uci.edu/ml/datasets/Letter+Recognition`

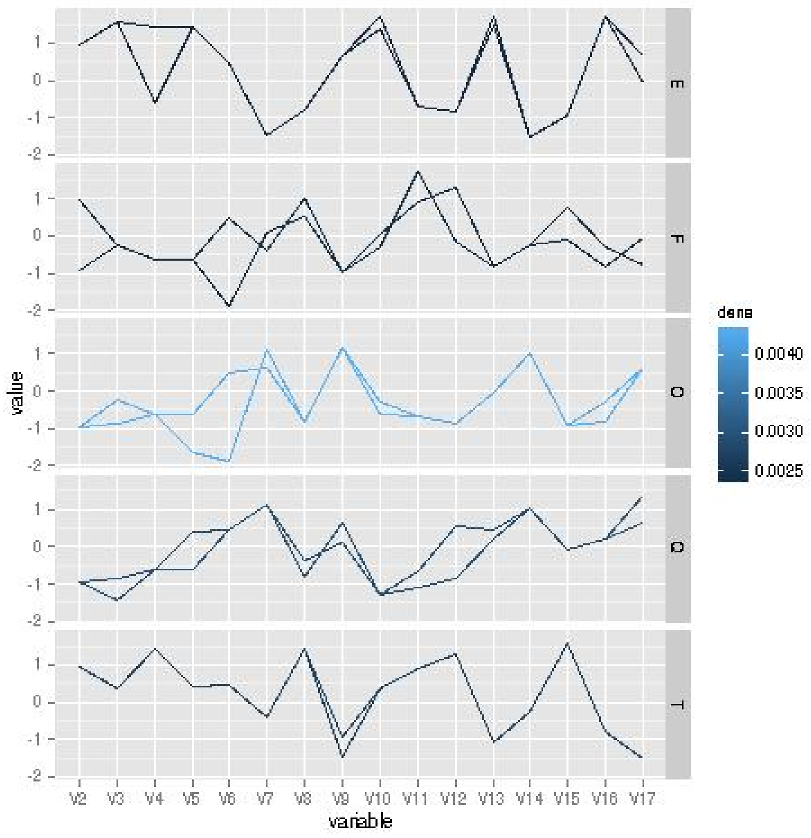

Figure 7: Letters data, selected letters

sphere and unit cube shrinks as p grows (Hastie *et al*, 2001). This is of course true, but by itself is not a cause for fundamental concern, because the property is metric-dependent. If for example we use the L_1 metric (Manhattan distance) instead of L_2 (standard Euclidean distance), the problem disappears.

This is not to say that Large p is not a problem. As noted earlier, Large-p will typically mean that standard errors are not minuscule, as noted earlier. Indeed, it has been long known that, for instance, in the case of exponential family models with p parameters, we should keep $p < \sqrt{n}$ to even assure our estimates are consistent (Portnoy, 1968).

Yet today $p >> n$ is commonplace.

Many variations of this problem have been studied, with interesting mathematical results. For instance, the principal components analysis (PCA) problem has been studied by some first-rate mathematical statisticians, under various assumptions (Jung and Marron, 2009). For example, some results assume "spikiness" of the population covariance matrix eigenvalues, meaning that a few eigenvalues are dominant.

And yet the dependence of these results on restrictive assumptions is troubling. Unlike the case of most of classical statistics, the effects of violations of these assumptions has not be well studied.

4.1 A Different Approach to the COD Problem

One of the most intriguing manifestations of the COD is that as p grows, the observations tend to become approximately equidistant from each other (Beyer *et al*, 1997) (Hall *et al*, 2005). This counterintuitive result, which will be made intuitive shortly, directly calls into question k-NN methods, and has more subtle but equally problematic implications for almost any type of regression analysis, for example. And yet I will argue that this property may hold a hint as to a road to resolving the COD problem.

But first, what is behind this distance paradox? Suppose for simplicity that the p components of a random variable X having the distribution of our population are i.i.d., and let $D = (D_1, ..., D_p)$ be the difference vector between two random members of the population. The squared distance between the two memmbers has mean na and variance nb for some a and b.[6]

The key point then is that the coefficient of variation of $||D||^2$, $a/(b\sqrt{n}) \to 0$ as $p \to \infty$. In other words, for large p, D is approximately constant.

But that reasoning also suggests a possible remedy. Consider a (squared) metric with positive weights w_i,

$$d(u,v) = \sum_{i=1}^{p} w_i(u_i - v_i)^2 \tag{3}$$

Our previously-calculated mean and variance of squared distance now become

$$a \sum_{i=1}^{p} w_i \tag{4}$$

and

$$b \sum_{i=1}^{p} w_i^2 \tag{5}$$

If we choose the w_i so that $\sum_{i=1}^{p} w_i < \infty$, then the coefficient of variation does <u>not</u> go to 0 as $p \to \infty$, and we don't have the approximate-equidistance problem.

Methods based on k-NN will then work fine, and the idea could be extended to other common statistical procedures.

There remains the question, of course, as to how to choose the weights. Actually, we could just use the geometric sequence 1, 1/2, 1/4... in all cases, but the real question is which components of X, i.e. which of our variables, to weight less heavily and which to put large weight on. I don't offer a definitive solution, but I believe in many cases the statistician will have at least a rough ordering of the variables in mind, which is all that is necessary.

5. Acknowlegements

The author would like to thank Noah Gift, Marnie Dunsmore, Nicholas Lewin-Koh, and David Scott for helpful discussions, and Hao Chen and Bill Hsu for use of their high-performance computing equipment.

REFERENCES

Artero, A., Ferreira de Oliveria, M. and Levkowitz, H. (2004), "Uncovering Clusters in Crowded Parallel Coordinates Visualizations," *Proceedings of the IEEE Symposium on Information Visualization*.

Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999), "When Is "Nearest Neighbor" Meaningful?," *Proceedings ICDT '99 Proceedings of the 7th International Conference on Database Theory*, 217-235.

Davidian, M. (2013), "Aren't *We* Data Science?" *Amstat News*, July 2013.

[6]The quantity a is double the population variance of a component of X, and b is an expression involving the fourth moment of that component.

Fan, T.H, Lin, D. and Cheng, K.F. *2007): “Regression Analysis for Massive Datasets, *Data and Knowledge Engineering*, 61, 554-562.

Garcia, V., Debreuve, E. and Barlaud, M. (2008), *Fast k Nearest Neighbor Search using GPU*, *arXiv*, arXiv:0804.1448.

Guha, S., Hafen, R., Kidwell, P. and Cleveland, W. (2009), “Visualization Databases for the Analysis of Large Complex Datasets,” *Journal of Machine Learning Research*, 5, 193200.

Hall, P., Marron, J. S., and Neeman, A. (2005), “Geometric Representation of High Dimension, Low Sample Size Data. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67, 3, 427444.

Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning*, Springer.

Jung, S. and Marron, J.S. (2009). “PCA consistency in High dimension, low sample size context,” *Annals of Statistics*, 37, 4104-4130.

Li, Shengqiao (2013), *FNN: Fast Nearest Neighbor Search Algorithms and Applications*, The Comprehensive R Archive Network, `http://cran.r-project.org/web/packages/FNN/index.html`.

Matloff, N. (2013), “Software Alchemy: Turning Complex Statistical Computations into Embarassingly-Parallel Ones,” under editorial review.

Portnoy, S. (1968), “Asymptotic Behavior of Likelihood Methods for Exponential Families When the Number of Parameters Tends to Infinity, Stephen Portnoy,” *Annals of Statistics*.

Unwin, A., Theus, M. and Hofmann, H. (2006). *Graphics of Large Datasets: Visualizing a Million*, Springer, 2006.

Wegman, E.J. (1992), “The Grand Tour in k-Dimensions,” *Computing Science and Statistics*, 22, 127-136.

Wickham, H. (2013), “Bin-Summarise-Smooth: A Framework for Visualising Large Data, 2013, submitted to Infovis 2013.