ABSTRACT STATISTICAL COMPUTING

R.W. Oldford
Department of Statistics and Actuarial Science
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

1.0 Beyond the calculation paradigm

A new paradigm for statistical computing is evolving which could significantly affect how statistical computing - and hence statistical analysis - will be done in the future. It is a subtle departure from an earlier *calculation paradigm* which holds that the rôle of statistical computing is to provide accurate calculations of mathematically specified quantities. Since these mathematically specified quantities are in fact meaningful statistical concepts (like estimators, summary statistics, probabilities, et cetera), the implemented algorithms can be regarded as software representations of the corresponding concepts. This different emphasis is the essence of the new paradigm. Namely, the rôle of *abstract statistical computing* is the accurate software representation of statistical concepts. (Abstract statistical computing is used here to emphasize the departure from the calculation paradigm.)

To illustrate, consider the case where a statistical parameter θ is to be estimated on the basis of observed data x_0. An estimator $T(x)$ will be used to give the estimate $T(x_0)$. The calculation paradigm requires that $T(x)$ be implemented in software, as $S(T(x))$ say, in such a way that the *evaluation error*, $| S(T(x_0)) - T(x_0) |$ (or a relative version of this), is reasonably small. If so, then $S(T(x_0))$ will be a reasonable representation for $T(x_0)$. The traditional task of statistical computing is to develop the theory, algorithms and software which will ensure a small evaluation error for a given $T(x_0)$ at all realizable x_0.

Now if $S(T(x_0))$ is a reasonable representation of $T(x_0)$ for all realizable x_0, then $S(T(x))$ should be a reasonable representation of $T(x)$. That is, the software is a representation of the *estimator*. If so, then we might expect to be able to do more with $S(T(x))$ than simply evaluate it. In particular, whatever manipulations of $T(x)$ are important (e.g. differentiation, integration, composition with other functions) should be performable on the representation $S(T(x))$. So, for example, a reasonable representation of $T'(x)$ should result from differentiating $S(T(x))$. Ideally, these representations will be such that the user manipulates the software representations *as if they are* the statistical concepts. How to achieve this illusion is the task of abstract statistical computing.

Clearly, this demands a different standard of plausibility for a representation. If, as in the above example, $T'(x)$ will also be used regularly then the error $S(T(x)) - T(x)$ would have at least three components: the evaluation error of $S(T(x))$ for all x, the evaluation error of its derivative (as a representation of $T'(x)$), and a measure of how easily $S(T(x))$ can be differentiated (to better affect the illusion). In general, the error will be multidimensional and will depend on both the concept being represented and the purposes of the representation.

A classic area of statistical computing which does not follow the calculation paradigm but which does fit the new paradigm is the generation of pseudorandom numbers. For example, a linear congruential generator is certainly implemented according to a mathematical specification and it can be implemented without error. However, the linear congruential sequence it produces is of no statistical interest unless it can be taken as a reasonable representation of a stochastic sequence. The implemented generator is then a software representation of a stochastic process.

Because there are many ways that the sequence could depart from randomness, the relevant error of this representation is inherently multidimensional. Moreover, which departures are important, and how great the error can be, will depend on the reason for using the generator. Perhaps most interesting, is the fact that it is known a priori that the error involved in such a representation can never be zero (since a deterministic mechanism is being used to model a stochastic one). Nevertheless, the error can be small enough that for most purposes the representation will be good enough.

In some sense, then, this new paradigm is not so new after all. Certainly any area which fits the calculation paradigm also fits this broader one. Moreover, areas of statistical computing (like pseudo-random number generation) which never followed the calculation paradigm always implicitly followed the abstract statistical computing paradigm. However, by explicitly adopting the new paradigm, many new problems arise whose solutions could produce some very powerful software tools for statistical analysis. In what follows, examples are selected from recent research in statistical computing which illustrate this point.

2.0 Abstract Statistical Computing

Figure 1 below illustrates the key problem of abstract statistical computing.

Statistical	$\rightarrow$	Software
Concept	$\leftarrow$	Representation

Figure 1.

There is a statistical concept which could be productively represented by software. The objective is to produce representations of these statistical concepts which are convincing enough to be manipulated as if they were the concepts.

Figure 1 is drawn with arrows in both directions (rather than as approximate equality) to indicate that the relationship between the two is not a static one. For example, it often happens that the definition of what the concept must entail changes (or equivalently, the kinds of error considered important change). The representation must be updated to reflect new attributes of the concept. Thus the concept determines the representation (the left to right arrow). Similarly, the right to left arrow indicates that the converse is also true - the representation can also determine the concept. By this I mean that shortcomings in the original concept are often more easily recognized once the representation is implemented and exercised. Then changes to the software are effectively clarifications of the concept.

The examples which follow illustrate the application of abstract statistical computing to many different topics in statistics. While not an exhaustive set, the software representations that have been implemented include linear transforms, statistical data, data features, statistical graphics, statistical models, statistical analyses, and statistical strategy. Each example is presented as briefly as possible - enough to identify the concept and the representation being considered. (the original papers should be consulted for more detail). Taken together these examples should give the flavour of the kinds of problems (new and old) to be addressed by abstract statistical computing.

2.1 Linear Transformations

This is an example of an old problem that has long been solved according to the calculation paradigm. The problem is to find the solution x, to the vector equation

$$Ax = b \tag{1}$$

where $A : k \times k$ of full rank and $b : k \times 1$ are known. The mathematical solution is clearly

$$x = A^{-1}b \tag{2}$$

Unfortunately, because of problems of numerical instability, this is not the solution with minimal evaluation error. That is, it is not numerically sound to solve (1) by first computing the inverse of the matrix A and then multiplying b by A^{-1}. Rather the solution (2) is arrived at by applying a series of linear transformations to both sides of equation (1) (say back substitutions from an LU decomposition of A). Furthermore, the best series of transformations to apply and/or how they should be applied may depend on what else is known about the matrix A (sparse?, bidiagonal?, triangular?, ...).

But the person who formulated the problem (1) knows the solution is given by (2) and wants that solution without worrying about any intermediate details. The level of abstraction the user is working at is completely specified by equations (1) and (2). Recognizing this, McDonald (1987) has suggested that the software also be usable and accurate at the same level of abstraction.

In particular, (1) can be described as the application of a linear transform A to the vector x yielding the vector b. And (2) is the application of the inverse transformation of A to the vector b to yield the solution x. There are four separate concepts here: a vector, a linear transform, the inverse of a linear transform (also a linear transform), and the application of a linear transform to a vector (as well as to other linear transforms but this can be ignored here). Hence there will be a representation for each.

The key idea of McDonald's (1987) approach is that a linear transform is of interest because it can be *applied* to a vector. The inverse, A^{-1}, of the linear transform A, is simply the sequence of transformations that are necessary to apply to A to yield the identity transform I. Hence it can be represented as such. Then, as far as the user is concerned, she has A^{-1} even though it was never explicitly calculated. The evidence is simply that A^{-1} has the expected behaviour when applied to a vector.

Moreover, the linear transforms could be typed according to their structure (triangular, sparse, et cetera). The inverse function would be implemented so that it recognized each type and produced the best series of transforms for the inverse of each. Again the user can be completely ignorant of this fact.

The evaluation error remains as small as before, but the user no longer needs to worry about such detail. She can happily operate at the more comfortable level of abstraction given by (1) and (2). The illusion is created that manipulating the software representations given by McDonald (1987) is equivalent to manipulating the mathematical symbols A, A^{-1} and b.

This is an example where the solution to an old problem can be improved by attacking it from a different perspective. However, it need not end there. As McDonald (1987) also points out, the concepts of linear transforms and vectors are not restricted to operate only on real vectors. Why not take an even more general view of these and build software representations for them?

2.2 Statistical Data

What is statistical data? The simplest statistical observation is the value of a variate recorded on some item or individual. As such, it contains a good deal more information than its value alone would indicate.

For example, irrespective of the individual involved, 11.4 centimetres is a suspicious value for the height of a human. This is because the variate (human height) together with the units of measurement (centimetres) give information that makes the value suspect. Apart from the actual measurement, both the individual on which the measurement is taken and the variate measured provide extra information which can be critical to the statistical analysis.

This suggests that an observation is actually composed of three roughly separable concepts (the individual, the variate and the measurement) each of which could be given its own software representation. This approach is adopted and described in greater detail in Oldford and Peters (1986a) and in Oldford (1987a). There, an object-oriented approach is used to define three classes: an **Individual** class, a **Variate** class, and a **Datum** class. Any particular observation is then represented as a composition of one member from each of the three classes.

The information available on a given observation is separated according to its most relevant source (individual, variate or measurement) and then attached to the software representation of that source (an **Individual**, **Variate** or **Datum**). To achieve this, every member of the **Individual** class has one "slot" where a label can be stored, and another where notes of arbitrary length can be stored. A facility (or "method") called "Edit-Notes" allows the user to change the information stored on the notes slot at any time. Any member of the **Variate** class also has these properties. Additionally, every member of **Variate** has a slot called "Range" where the natural range of the measurements is recorded (e.g. a numerical range for a continuous variate, a list of possible values like "red" "white" or "gold" for a categorical variate). Similarly, each member of the **Datum** class has three slots: "DataValue" which contains the measurement (a number or a string), "Censoring" (None, left, or right), and "SignificantDigits" which records the number of significant digits in the actual measurement (where appropriate).

To form an observation, these three components are combined in a variety of ways. A multivariate observation for example is the composition of a single **Individual** and many **Variate-Datum** pairs. Similarly, a batch of measurements is the composition of a single **Variate** and many **Individual-Datum** pairs.

An important advantage to these representations is that information is shared in the software exactly as it is in data. For example, suppose two different variates are measured on the same person and two batches are consequently constructed so that the same person appears in both. The software representation of that person will be a unique **Individual** and the software representation of each batch will contain a pointer to this unique **Individual**. That is the individual information is not repeated. Similarly, multivariate observations share access to unique **Variates**. Whole **Individual-Variate-Datum** triples can be shared between multivariate observations and batches.

Like their component parts, there will be information available on multivariate observations (e.g. the observation appears to be very different from others on the same variates) and batches (e.g. the circumstances under which the data were collected) which could be usefully attached to their representations. Hence these representations (**Cases** and **Batches** respectively) also have slots for notes. Further, there is statistical information, like the median and quartiles, in a batch of numbers. Hence, the means of extracting such information are also part of the representation **Batch**. Similarly, other kinds of statistical information are relevant to, and hence should be extractable from, a **Case**.

Thus, from the representations of simple statistical data concepts more complex data structures are built by composition. These in turn represent more complex but familiar data concepts. At each new level, the representations are refined to better reflect the concepts. Other data structures and examples are given in Oldford (1987a). A similar approach was adopted by McDonald (1986).

Unlike linear transforms, the concepts for statistical data information did not have precise definitions prior to their representation. Rather, they were (and continue to be) refined by exercising the representation and making appropriate changes to the software.

2.3 Data Features

There are many ways to describe a batch of numbers. It could be symmetric or asymmetric, thin-tailed or heavy-tailed, unimodal or multimodal, have outliers or not, be quite gaussian looking or not, and the list can go on. These features are not unrelated. A batch cannot be both asymmetric and gaussian, for example.

Gale and Lubinsky (1986), in a system called TESS, have proposed that such features be related in a tree as shown (partially) in Figure 2. Figure 2. Data features as a tree.

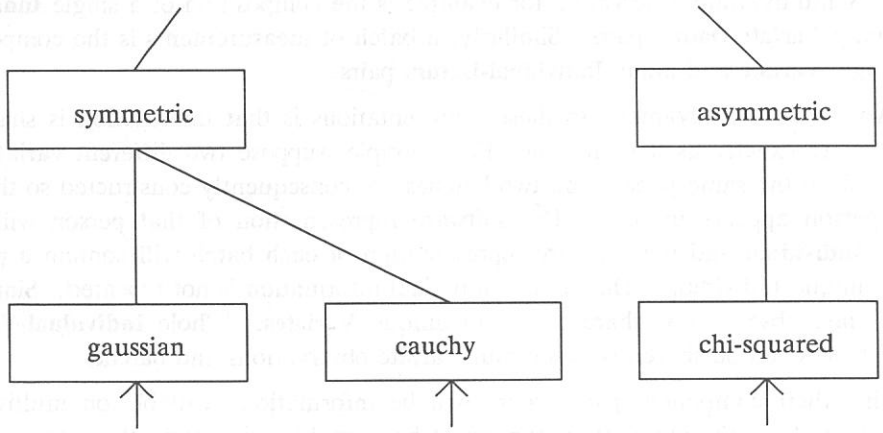

Figure 2. Data features as a tree.

Here the features "asymmetric" and "symmetric" are mutually exclusive and hence appear as separate branches. Similarly, because a batch cannot be either gaussian or cauchy unless it is also symmetric, the "gaussian" and "cauchy" features are separate branches below "symmetric".

While the tree represents the connections between features, the nodes are meant to represent the individual features. Consequently, a good deal of information is attached to each node. In particular, tests (statistical or otherwise) for the presence or absence of a feature are accessible to each node, as are example data sets that have the feature present or absent to various degrees. These examples can be displayed to the user to indicate the nature of the feature being investigated.

In this way the concepts (the data features) are clarified by the software representation. For example, gaussianity is an easily and precisely defined feature of *a probability distribution*. However it is a much more nebulous feature of *a batch of numbers*. Defining the feature tree and each node as outlined above is one way to convey what is meant by a gaussian batch - "These tests must be passed, here are some examples of what is meant, and the batch must also be symmetric."

This illustrates how software can be used to clarify (or at least more precisely communicate) what is meant by an informal statistical concept. A representation gives a more formal basis for studying the concept. Improvements to the representation (and hence refinements of the concept) can be based on how it falls short of our understanding of the concepts involved. In the present example, a collection of feature networks as in Figure 3, would seem to be a more accurate representation of the relationship between statistical features (Oldford 1987b).

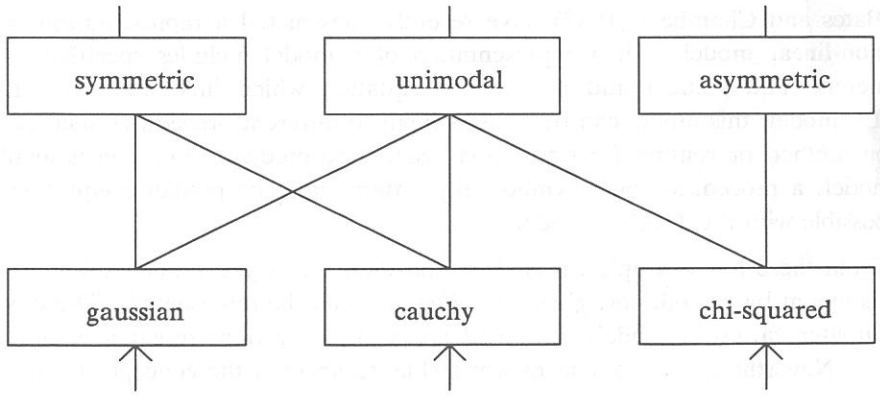

Figure 3. A feature network.

2.4 Statistical Models

Models are a very basic element of statistical analysis and hence could be very usefully represented in software. Indeed, a variety of representations have long been implemented in statistical packages. Considering a few here will give some idea of the range

of possibilities.

The simplest approach is to have the model specified implicitly by the procedure used to estimate its unknown parameters. This is the approach taken in S (Becker and Chambers, 1984). There the linear model, for example, is implicitly specified by its estimation: reg(x,y).

GLIM, an interactive package to fit generalized linear models (Numerical Algorithms Group Ltd., UK), requires the model to be defined explicitly. The model has several components: a stochastic component, a systematic component (the linear predictor), a link function between the two, and of course the variables which may appear in the model. In GLIM these are specified and the model is fitted. Modelling proceeds by changing one of these components, typically the linear predictor. At any time in the analysis, the user has only one model available - the one currently fitted (possibly also the immediately previous model).

By contrast, TROLL, an interactive econometric modelling system (from the Massachusetts Institute of Technology), allows for the possibility of many models coexisting simultaneously (of course at the time of parameter estimation the particular model must be identified). A TROLL model is a separate data structure having many components. Exogenous and endogenous variables, parameters, stochastic error terms, and equations relating them, are all part of the model. The TROLL model can be estimated in a number of ways, can be simulated for given parameter values, and can be editted at any time.

Bates and Chambers (1987) have recently constructed a representation of a general non-linear model. Their representation of a model includes specified variables, parameters (and values) and a predictor equation which links the two. Like the TROLL model, this model can be used as input to different procedures such as an estimation method or routine for simulation. Since the predictor equation is available in the model, a procedure could symbolically differentiate the predictor equation (this is also possible with the TROLL model).

From these few examples, it is clear that there is, as yet, no general consensus on what is meant by a model or, given one, how it should be represented. There would be even greater variety if models in related areas like operations research were also considered. Nevertheless, it does seem worthwhile to abstract the concept of a model and give a general representation for one. Functions written to operate on a general model are clearer and simpler. Bates and Chambers (1987) give the bootstrap resampling procedure as an example. Their bootstrap function takes the model, an estimation procedure and a set of sample data sets as input. The bootstrap function is then trivial - it estimates the model for each data set in the set of sample data sets. By comparing and contrasting implemented representations, more generally held views of the components of a model might result.

2.5 Analysis Management

There has been a good deal of interest lately in providing software tools to help the statistician manage his analysis (e.g. Nicholson et al 1984, Becker and Chambers 1986, Oldford and Peters 1986b). Any tool for managing an analysis presupposes some model of how an analysis proceeds, or equivalently a model of an analysis.

In Oldford and Peters (1986b), an analysis is modelled as a collection of directed acyclic networks. These are displayed in a window on the workstation's screen as shown in Figure 4.

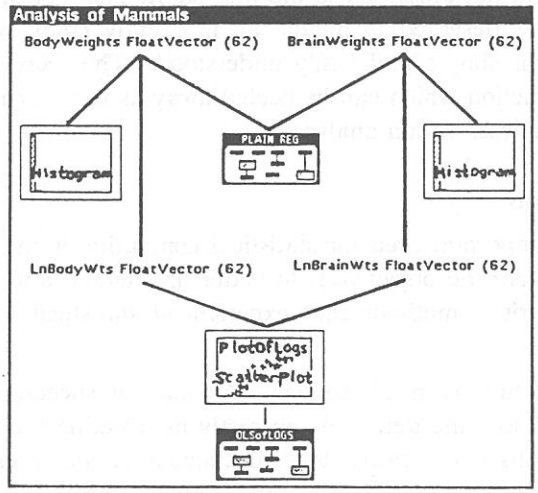

Figure 4. An Analysis Map.

Each model represents a step in the analysis and is implemented as an object that is statistically meaningful. The interface provides a menu at each node (by selecting the node with a mouse pointing device) which gives access to the actions that are usually taken at that step. The logical flow of the analysis is indicated by the links between nodes.

In Figure 4, the analysis began from two data vectors (Body Weights and Brain Weights) of 62 observations each. A histogram of each data vector was examined, the logarithm of each data vector taken and then the logged data (LnBodyWts and LnBrainWts) were plotted. The remaining nodes, PLAIN REG and OLSofLOGS, represent subanalyses whose detail has been suppressed from the display. All such detail can be retrieved by selecting the "Zoom" item from the menu on the node of interest. Zooming on a data vector produces a structure that shows the actual data values, Zooming on a plot produces the plot, and Zooming on a subanalysis produces a new display that is like Figure 4 in every way except for different contents. Since one analysis can contain another, this model of an analysis allows for arbitrarily many levels of detail.

The model of an analysis also tracks more than one kind of link. It can follow three kinds: user determined "analysis" links, and programmatically determined "causal" and "data-flow" links. (See Oldford and Peters 1986b for more detail).

The model seems to work well but has yet only been tested on rather simple analyses. More complex analyses will be required before the limitations of the design are determined.

Even so, two general lessons can be drawn. The first is that the user interface can be critical in establishing the credibility, and hence usefulness, of the software representations. Second, (and not unrelated to the first point) this representation is an example where the software is designed to handle an inherently complex problem yet present itself in a way that is simple and easily understood. The secret, it would seem, is to build layers of abstraction which can be peeled away as the user requires more detailed information (e.g. analyses within analyses).

2.6 Statistical Strategy

A potentially very important area for statistical computing is the representation of statistical strategies. Here the objective is to better understand and perhaps capture some of the informal heuristic methods that experienced statistical analysts find useful in practice.

A principal difficulty with this endeavour is that the successful application of these methods will depend to some degree on correctly interpreting the substantive context of the problem. And this is not easily done mechanically. For those methods where the context matters little (e.g. an adhoc method for identifying data patterns), a reasonably accurate software representation can probably be built. If interpreting the problem context is critical to the method (e.g. general methods of modelling), then its representation must be undertaken with care. For further discussion on context and strategy see Oldford and Peters (1986c), Mallows and Pregibon (1987) and Oldford (1987b) (and references therein). A number of papers on statistical strategy in general can be found in Gale (1986).

A number of approaches to implementation have been undertaken. Gale and Lubinsky (1986) set out some of the elements a computer language must have to express statistical strategies. (See also Oldford 1987b which discusses this paper in more detail). In the same paper, they also suggest that the feature trees of section 2.3 be used to represent the features of a strategy. The tree of Figure 2 would represent a strategy for describing the shape of a unimodal batch of numbers. The strategy then amounts to a tree search (see also Pregibon, 1986). The analysis management model described in Section 2.5 can be used to represent statistical strategy in cases where the context is critical. In such cases, it is left to the user to interpret the context and consequently decide what action to take next - only context independent guidance is provided by the software. However, at an individual node of the analysis the context is defined narrowly enough that some low level strategies could be implemented there.

(For further discussion see Oldford and Peters 1985, 1986b,c).

Statistical strategies may never be completely captured by software representations. However, by implementing some aspects of different strategies, they will in time become better understood.

2.7 Statistical Graphics

A very active and important area, interactive statistical graphics has recently made many gains by adopting the paradigm. Basically, a number of statistical graphical concepts can be represented separately and combined to create more complex graphics. As was the case with statistical data, an important advantage to the approach is that components of graphics can be shared. The components to be shared range from the data being viewed (McDonald 1986, Becker and Cleveland 1987, Stuetzle 1987), to operations like projections in a pipeline of views (Buja et al 1986, Hurley 1987). Such sharing makes it reasonably straightforward to achieve linking - for example, between different plots and between plots and the data they view. For further detail, the reader is referred to the above-mentioned articles and the references they contain.

3.0 Concluding Remarks

The above examples by no means exhaust the range of application of abstract statistical computing. Nor should the length of discussion devoted to any one topic be taken to construe its importance (this would be a gross error, for example, in the case of statistical graphics). Rather, it is hoped that these few examples give the flavour of the general paradiagm.

In closing, it must be pointed out that the important issue of efficiency does not fall outside of the paradigm. If machines had infinite memory (and zero access time) and infinite calculation speed, efficient code would be a non-issue. But why? It would be because the machine can calculate and access memory quickly enough that the user can manipulate the representations as quickly as he can manipulate the concepts. If he could not, the illusion that the representation was the concept would be impaired. Since such machines do not exist, efficiency of calculation and memory management, like the interface, are important to make the representation believable.

Bibliography

Bates, D.M. and J.M. Chambers (1987). "Statistical Models as Data Structures", *Statistical Research Report No. 42*, AT&T Bell Laboratories, Murray Hill, N.J.

Becker, R.A. and J.M. Chambers (1984). *S: An Interactive Environment for Data Analysis and Graphics.* Wadsworth Inc., Belmont CA.

Becker, R.A. and J.M. Chambers (1986). "Auditing Data Analyses", *Proc. of the ASA: Statistical Computing Section.*

Becker, R.A. and W.S. Cleveland (1987). "Brushing Scatterplots", *Technometrics, 29*, pp. 127-142.

Buja, A., C. Hurley and J.A. McDonald (1986). "A data viewer for multivariate data", *Computer Science and Statistics: Proc. of the 18th Symposium on the Interface*

Gale, W.A. (editor) (1986). *Artificial Intelligence and Statistics*. Addison-Wesley, Reading MA.

Gale, W.A. and D.J. Lubinsky (1986). "A Comparison of Representations for Statistical Strategies". *Proc. of the ASA: Stat. Comp. Sect.*

Hurley, C. (1987). "The Data Viewer: A Program for Graphical Data Analysis" unpublished doctoral dissertation, University of Washington, Seattle WA.

Mallows, C. and D. Pregibon (1987). "Principles of Data Analysis", *Bulletin of the I.S.I. (this volume)*

McDonald, J.A. (1986). "Antelope: data analysis with object-oriented programming and constraints", *Proc. of the ASA: Stat. Comp. Sect.*.

McDonald, J.A. (1987). "CACTUS", presented at *Computer Science and Statistics: 19th Sym. on the Interface*

Nicholson, W.L., D.B. Carr, P.J. Cowley, and M.A. Whiting (1984). "The Role of Environments in Managing Data Analysis", *Proc. of the ASA: Stat. Comp. Sect.*, pp. 80-84.

Oldford, R.W. (1987a). "Object-Oriented Software Representations for Statistical Data". *STAT-87-18*, University of Waterloo, CANADA

Oldford, R.W. (1987b). "A discussion of two papers on 'Applying Artificial Intelligence Programming Techniques in Statistics' " *STAT-87-17*, University of Waterloo, CANADA

Oldford, R.W. and S.C. Peters (1985). "DINDE: Towards More Sophisticated Software Environments for Statistics". *SIAM Journal of Scientific and Statistical Computation* (to appear 1988).

Oldford, R.W. and S.C. Peters (1986a). "Object-Oriented Representations of Statistical Data". *COMPSTAT-1986*, pp. 301-306.

Oldford, R.W. and S.C. Peters (1986b). "Data Analysis Networks and DINDE". *Proc. of the ASA: Stat. Comp. Sect.* pp. 19-24.

Oldford, R.W. and S.C. Peters (1986c). "Implementation and Study of Statistical Strategy". pp. 335-353 in Gale, 1986.

Pregibon, D. (1986). "Data Analysis and Search", *Statistical Research Report No. 29*, AT&T Bell Laboratories.

Stuetzle, W. (1987). "Plot Windows", *JASA, 82*, pp. 466-475.

Summary

A new paradigm for statistical computing that has been evolving is identified. The paradigm of abstract statistical computing is to use software to model (possibly abstract) statistical concepts. Some examples from recent statistical computing research are discussed to illustrate the paradigm.

Résumé

Nous présentons un nouveau paradigme provenant du calcul statistique par ordinateur. Pour ce paradigme de calcul statistique abstrait, nous utilisons un logiciel pour modeler des concepts statistiques possiblement abstraits. Nous discutons enfin quelques exemples de recherche récente dans ce domaine, pour illustrer ce paradigme.