National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

The effect of sampling error on the interpretation of a least squares regression relating phosphorus and chlorophyll.

by

David C. Beedell

A thesis presented to the Faculty of Graduate Studies of McGill University in partial fulfilment of the requirements for the degree of Master of Science.

Department of Biology
McGill University
Montréal, Québec
Canada

February 1995

TABLE OF CONTENTS

TABLES AND FIGURES

Tables

Figures

ABSTRACT

Least squares linear regression is a common tool in ecological research. One of the central assumptions of least squares linear regression is that the independent variable is measured without error. But this variable is measured with error whenever it is a sample mean. The significance of such contraventions is not regularly assessed in ecological studies. A simulation program was made to provide such an assessment. The program requires a hypothetical data set, and using estimates of S^2 it scatters the hypothetical data to simulate the effect of sampling error. A regression line is drawn through the scattered data, and SSE and r^2 are measured. This is repeated numerous times (e.g. 1000) to generate probability distributions for r^2 and SSE. From these distributions it is possible to assess the likelihood of the hypothetical data resulting in a given SSE or r^2. The method was applied to survey data used in a published TP-CHLa regression (Pace 1984). Beginning with a hypothetical, linear data set (r^2=1), simulated scatter due to sampling exceeded the SSE from the regression through the survey data about 30% of the time. Thus chances are 3 out of 10 that the level of uncertainty found in the surveyed TP-CHLa relationship would be observed if the true relationship were perfectly linear. If this is so, more precise and more comprehensive models will only be possible when better estimates of the means are available. This simulation

approach should apply to all least squares regression studies that use sampled means, and should be especially relevant to studies that use log-transformed values.

RÉSUMÉ

La régression linéaire simple est un outil commun dans la recherche écologique. L'une des suppositions centrales de la régression linéaire simple est que la variable indépendante est mesurée sans erreur. Cependant, la variable indépendante est mesurée avec erreur lorsqu'il s'agit d'une moyenne d'échantillon. La signification de telles infractions n'est pas régulièrement évaluée dans les études écologiques. Un programme de simulation a été créé dans le but de fournir une telle évaluation. Le programme exige un ensemble de données hypothétique et, en utilisant des calculs approximatifs de S^2, disperse les données hypothétiques afin de simuler l'effet d'erreur d'échantillonage. Ensuite, une droite de régression est établie à travers les données dispersées, et SSE et r^2 sont calculés. Ce processus est répété nombreuses fois (e.g. 1000) afin de produire des distributions de probabilité pour r^2 et SSE. À partir de ces distributions il est possible d'évaluer la probabilité que les données hypothétiques résultent dans un SSE ou r^2 donné. La méthode a été appliquée à une régression TP-CHLa publiée (Pace 1984). À partir d'un ensemble de données hypothétique linéaire (r^2=1), la dispersion dû à l'échantillonage simulée a excédee le SSE de la régression publiée 30% des fois. Ainsi, il y a 3 chances sur 10 que le niveau d'incertitude que l'on voit dans la relation TP-CHLa publiée aurait été

observée si la vraie relation était parfaitement linéaire. Cette approche de simulation devrait s'appliquer a tous les études de régression linéaire simple qui utilisent des moyennes d'échantillons, et devrait s'avérer particulièrement pertinente aux études qui utilisent une transformation de données logarithmique.

ACKNOWLEDGEMENTS

I am indebted to many, many people who have helped me. I am indebted to those who taught me that which they believed, and I am indebted to those who left their thoughts in places that I happened upon. I appreciate their struggles, and the kindness which is the source of all good teaching.

I am grateful to a teacher who stressed the significance of natural variation, and to one whose enthusiasm for important questions was infectious. I am grateful for the freedom that I had while pursuing graduate studies, and perhaps even for the false-starts, dead-ends, and indecision that came with that freedom. I thank my supervisor, others who gave me feedback, my friends, and every member of my family - each of you helped me in a special way.

My difficulties with forming the ideas in this thesis, and then finally expressing them, gave me a greater appreciation of how dependent we are upon each other - for guidance, for encouragement, for understanding. No scientific concept is entirely original, nor is born solely from one individual. I once thought otherwise.

Official sources of funding were an NSERC 1967 graduate scholarship, as well as an NSERC grant to R.H Peters. This study uses raw data kindly provided by Michael Pace.

INTRODUCTION

In many branches of science we must sample "populations" to estimate the quantities we wish to know. This is especially so in ecology, where models often represent large spatial and temporal dimensions. Yet, sampling introduces such uncertainty to all of our estimates that "true" values are almost never known. In turn, this uncertainty can dramatically influence the way we test our models. This study focuses on uncertainties in the estimates of mean values, and their consequences for simple linear regression studies. This is exemplified by scrutinizing one previously published regression equation. I did not attempt to investigate error from causes other than sampling, nor how it would interact with sampling error.

Whether our ultimate aim is to predict or explain, an important part of science is to consider the likely models that might be compatible with our data. This directs our science by narrowing the choice of models that we might investigate. In a simple regression study, an obvious model to consider is that there is an imperfect relationship between X and Y, whose slope is the slope of the regression equation. The degree of imperfection - how much explaining we have yet to do - is quantified by the r^2. Another model that almost always is considered is that there is no relationship between X and Y; this is usually the null

hypothesis for statistical tests of significance.

Once a predictive equation has been developed, and the null hypothesis discarded, the next step is usually to further investigate the hypothesis that the relation between X and Y is imperfect because other factors have influenced it. Generally, this involves residual analysis of some sort, which may lead to another regression study - one that includes more independent variables.

If we have used average values in our analysis, there is another model that should be considered before extensive studies are undertaken that require costly measurements of more variables. One must ask if the existing data set is compatible with a perfectly correlated mean X and mean Y. Few published ecological studies do so.

A perfect correlation between mean X and mean Y can be thought of as another null hypothesis - in addition to "no relationship between mean X and mean Y". The reason that we should consider it, even though the data clearly do not lie on a straight line, is that error results from sampling a population. When we sample a population to estimate the true average for that population, a random error is introduced into our data. Like a jumping bean, our measured mean strays from the true population average. If we can find a way to simulate these "data hops", we can test the hypothesis that the data are consistent with a perfect relationship.

The simulations of sampling error are easy in

principle. First, a perfectly fit relationship between mean X and mean Y is hypothesized (the regression line fit to the data is a reasonable choice). Then a sampling program is simulated by introducing random sampling error to the hypothetical values along the regression line. Finally, simulated data sets are compared with a survey data set (Figures 1a,b,c,d).

The remainder of the study discusses a computer program that performs such simulations. The approach is examined using a regression between mean total phosphorus (TP) vs. mean chlorophyll a (CHL) (Pace, 1984) from 12 bodies of water (lakes or lake basins) in the Eastern Townships of Quebec. This data set is interesting for several reasons.

TP-CHL relationships have been an important area of study for close to 30 years (Sakamoto, 1966; Vollenweider, 1968; Dillon and Rigler, 1974; Smith, 1982; Molot and Dillon, 1991), have proven a successful management tool (Molot and Dillon, 1991), and illustrate the benefits of "predictive ecology" (Peters, 1986). Yet, despite great effort spent looking for additional predictor variables (Smith, 1982; Canfield et al., 1983; Pace, 1984; Prairie et al., 1989), a good deal of residual scatter around these relationships awaits explanation (France et al., 1994). The high uncertainty associated with estimating mean phosphorus and CHL may be one explanation (Pepin, 1987; France et al., 1994). Pace's (1984) data set is interesting, therefore,

because it is one example from a body of important work, and conclusions about Pace's data set may be relevant to this larger body. It is also interesting because Pace (1984) did not consider uncertainty in mean TP and CHL when he investigated the residuals in his TP-CHL relationship. This oversight could have affected his interpretation of statistically insignificant correlations ("negative results"), and in at least one case his interpretation has been used as corroborating evidence (Morales-Baquero et al., 1994). Finally, McQueen et al. (1986) interpreted the residuals from several nutrient-consumer relationships without considering the effects of uncertainty in estimating means (Pepin, 1987). One of these relations was the Pace (1984) TP-CHL relationship.

METHODS

ASSUMPTIONS AND GOALS

The following conditions are assumed:

[1] Ordinary least squares linear regression is used on sample means from a survey. I will call these sample means "survey mean X" and "survey mean Y".

[2] The survey mean X values and survey mean Y values are imperfect estimates of the unknown "true" means. I call this

discrepancy "sampling error". To simplify this introductory study, I assume that all measurements are free of error. Sampling error, therefore, is entirely due to the stochasticity introduced by choosing to measure only some of the values that underlie a true mean. I also ignore the fact that the values used to estimate a sample mean might, themselves, be averages of several measurements (commonly triplicates are used). These simplifications allow sampling error to be more easily simulated, and are realistic when measurement error is small and the triplicate values do not differ much.

[3] The sample size (n) used to estimate each mean is known, as is the variance (S^2) associated with the values used to estimate a mean. I use the terms "replicate X" and "replicate Y" to describe the values used to calculate survey means. The following form of variance is used:

$$S^2 = \frac{\sum_{i=1}^{n} (\text{survey mean X} - \text{replicate } X_i)^2}{(n - 1)}$$

[4] The following is known about the regression analysis of the survey data: the number of data points is d; the slope is m; the intercept is b; the sum of squares (SSE) is the "survey SSE"; the r^2 is the "survey r^2".

[5] A perfect relation between X and Y is hypothesized that has d points each lying on the regression line from [4]: Y = m * X + b . These points are hypothesized to be the "true" means that the survey attempted to estimate. (I will refer to this set of d hypothetical values as H_1).

The goal of the thesis is to determine if H_1 is compatible with the survey results. I set two related tasks to achieve this goal: (1) I estimate the probability of finding an SSE greater than, or equal to the survey SSE, if H_1 is indeed correct; (2) I determine whether the survey r^2 is different in a statistically significant way from the r^2 = 1 associated with H_1.

These two tasks are accomplished by: (1) recording the SSE from 1000 simulated data sets, and then checking the percentage of times that these exceeded the survey SSE; (2) recording the array of r^2 values from 1000 simulated data sets, sorting that array, and then developing 95% CI by selecting the values at position 25 and 975.

As the probability in (1) increases, or when there is no statistically significant difference in (2), we should consider more seriously that H_1 might be correct, and in subsequent surveys shift the emphasis of research towards reducing sampling error, and away from seeking additional explanatory variables.

The approach required is similar to power analysis

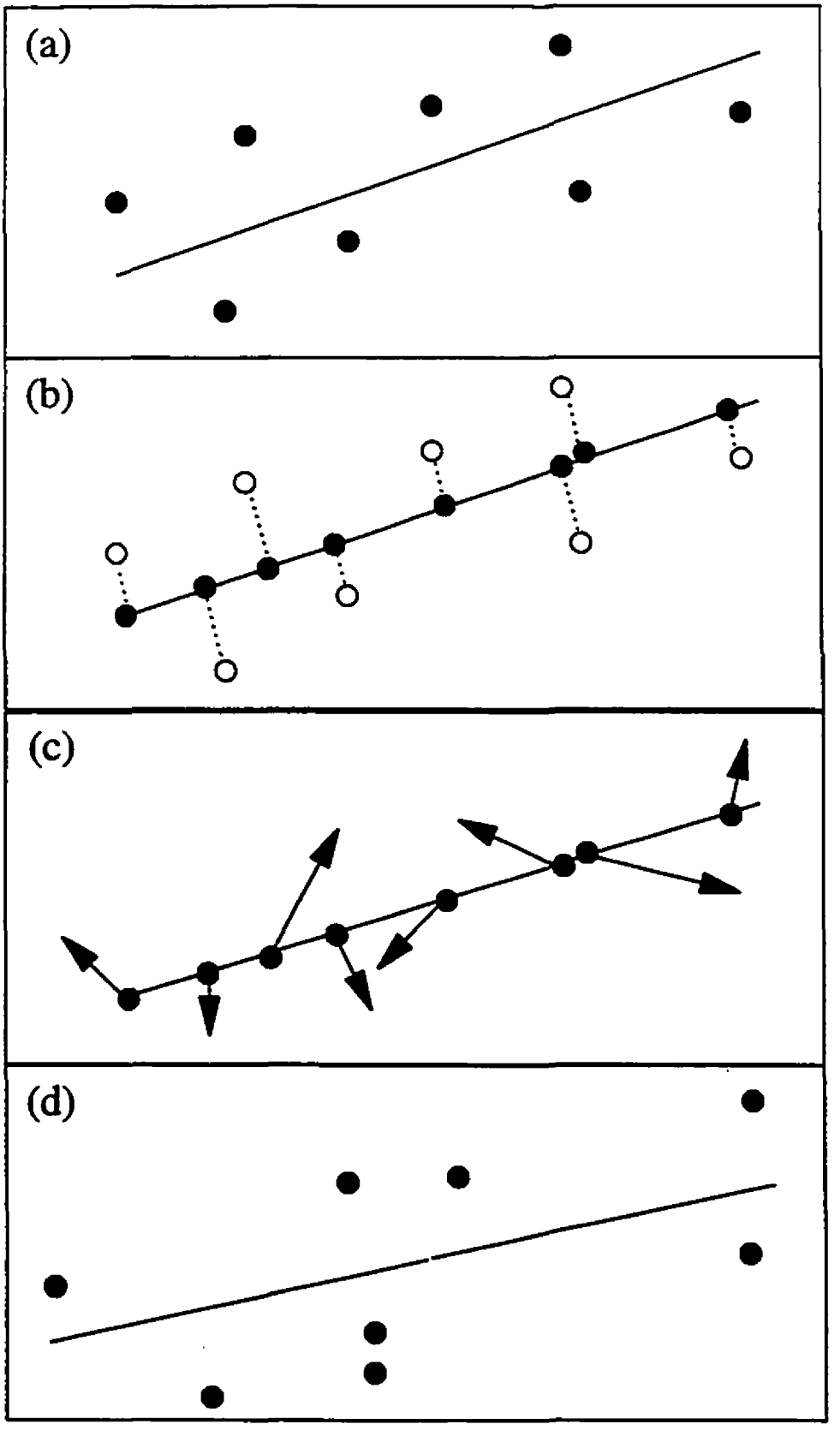
(Zar, 1984; Peterman, 1990) because it questions whether some estimated value such as the survey SSE or survey r^2 might come from a population with a different value, and answers this question by estimating the distribution of SSE or r^2 values that are possible given H_1. The approach differs from a power analysis in that no attempt is made to quantify a type II error.

The usual methods for estimating the distribution of regression statistics (eg. r^2, Zar 1984) are inappropriate because sampling error is in both X and Y. Even if error were restricted to Y, these convenient methods would be compromised when sampling error differs over Y, as it does in this study (and in many ecological surveys). Simulation is used, therefore, to estimate the distributions of SSE and r_2 given H_1.

OVERVIEW OF THE SIMULATIONS

The simulation requires that sampling error be introduced to each of the data points in H_1 (Figure 1). Each hypothetical data point has an X and Y component (hypothetical mean X, hypothetical mean Y), and the simulations assume that the sampling error in each component is independent.

FIGURE 1: A depiction of the simulation process. (a) The data set of means from a field survey, and the regression line through it. (b) Hypothetical XY coordinates (solid circles) are placed where each dotted line intersects the regression line. The survey data set (open circles) is shown for comparison. (c) Sampling error is simulated. (d) A regression line is fit to the simulated data set and SSE and r^2 are measured.


(a)
(b)
(c)
(d)

For each of the hypothetical mean values, a pseudo-random number is chosen from a Normal distribution with mean equal to the hypothetical mean and variance equal to S^2/n (S^2 is the variance of the replicate values used to estimate a survey mean, and n is the sample size used). The shape of these distributions is based on the Central Limit Theorem (CLT) (Hogg and Tanis, 1983). This theorem is convenient because it allows us to estimate the shape of a distribution of means without knowing exactly the shape of its underlying distribution. This reduces the time required to simulate each X and Y value (simulated mean X, and simulated mean Y) since mean values can be simulated directly, and need not be calculated from n simulated replicates.

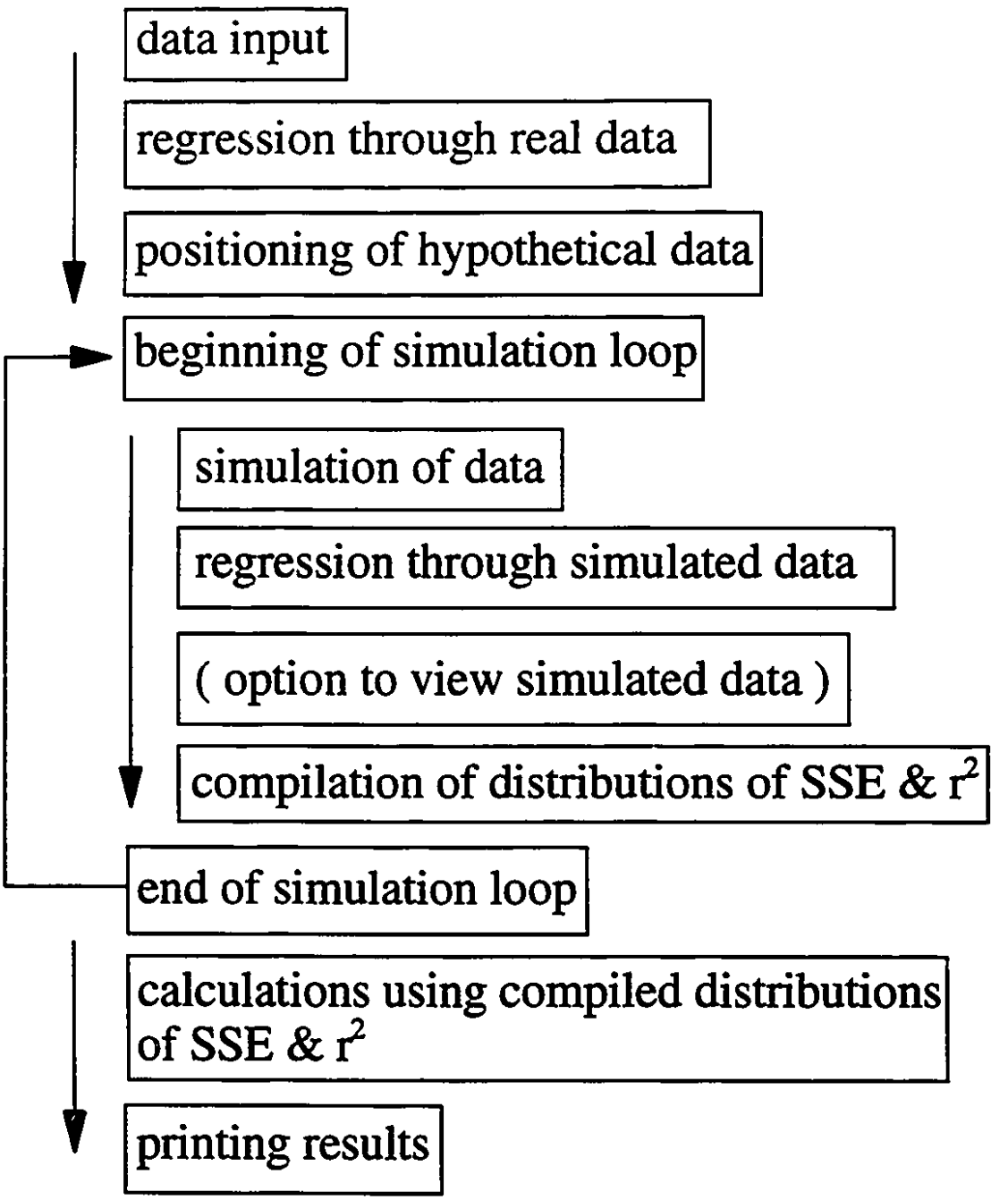
One simulated data set is complete once a simulated mean has been generated for each of the hypothetical means. The simulated mean Y values are then regressed on the simulated mean X values, and the SSE and r^2 are recorded (model SSE, and model r^2). This is repeated for 1000 simulated data sets to build a distribution of SSE and r^2 given H_1. Once those distributions are built, they can be compared with the SSE and r^2 from the survey data (survey SSE, and survey r^2). Figure 2 shows a simplified flow chart for the program, whose code is in Appendix II.

Visual examination of simulated data sets is also a way to check if the survey data is consistent with a perfect-fit H_1. If scatter about the survey regression line is due to

sampling error, then simulated data will resemble the survey data in a substantial proportion of the simulations (so long as the simulations are realistic).

FIGURE 2: The simulation program is written in TBASIC and accepts survey means, S^2 values, sample sizes, and min/max limits from file or keyboard, simulates sampling error, and fits regression lines. If desired, simulations can be viewed on a computer screen.

Flow chart for simulation program



CHOICE OF HYPOTHETICAL VALUES (H_1)

There is one hypothetical mean for every survey mean. If more than 5% of the hypothetical means fall outside the 95% confidence intervals (CI) around their corresponding survey means, the compatibility of the hypothetical data set and the survey data set would be in doubt - even if the model SSE were to frequently exceed the survey SSE in subsequent simulations. It is therefore desirable to choose hypothetical data that fall as close as possible to their corresponding survey data. That way, when hypothetical data are incompatible with the survey data, one can conclude that all linear hypothetical data sets are incompatible with that survey data, and reject the linear hypothesis.

The positions of hypothetical data were determined by projecting perpendicular lines from the survey regression line to the survey data (Figure 1a). Hypothetical data were placed where these perpendiculars touched the survey regression line. Perpendicular projections yield the shortest distance between the survey regression line and the survey data, but do not necessarily reveal the hypothetical data (along the survey regression line) most likely to be compatible with the survey data. Maximizing this compatibility requires minimizing the "statistical distance" (sensu Johnson and Wichern, 1982: p.20) between the hypothetical data and survey data. Perpendicular projections

were used because they were simpler, yet still provided hypothetical data that were compatible with Pace's (1984) survey data. The compatibility of the survey means and the hypothetical means was assessed using 95% CI based on a t-distribution. This distribution was used because the distributions of survey means were assumed to be Normal (using the CLT), and had an estimated variance.

SIMULATED MEAN VALUES

A logistic approximation method (pers. comm. D.Roff, Biology McGill University) was used to generate Normal distributions:

$$\text{simulated mean} = [.5513 * \log_e(\text{rand}/(1-\text{rand}))] * [(S^2/n)^{1/2}] + \text{hypothetical mean},$$

where the "hypothetical mean" above is not log-transformed; and "rand" is a single pseudo-random number from a uniform [0,1] distribution. This approximated the desired distribution with mean equal to the hypothetical mean, and variance equal to S^2/n. Since Pace (1984) $\log_{10}$-transformed his data before regression analysis, each simulated mean was also $\log_{10}$-transformed.

No simulated mean value was allowed to exceed the 95%

confidence limits of its hypothetical mean, nor fall beyond minimum (min) or maximum (max) values that were chosen for TP and CHL in each lake. These arbitrary min and max values provided biologically based limits to simulation, and helped offset the bias in simulations that might result if the S^2 value from Pace (1984) were inflated. I set min and max values as the smallest and largest replicate values. For instance, if the smallest replicate TP value in a lake were 4 mg/l, then the simulated mean TP for that lake would not drop below that value. The 95% CI also limited extreme simulations, and made simulations more realistic if the min or max values differed very much from the mean. These CI may have caused sampling error to be underestimated when min/max values were realistic, yet more lenient than the 95% CI. In general, all these limits restrict SSE and increase r^2, so they make these tests conservative.

TESTING ASSUMPTIONS AND SENSITIVITY TO PARAMETER FLUCTUATIONS

Distributions of model SSE and r^2 were generated using 1000 simulated data sets. I tested the appropriateness of this number then tested the simulation program's sensitivity to the following:

(1) the min and max values used to limit simulated means,

(2) the use of the CLT to approximate the shape of distributions,

(3) the S^2 estimates,

(4) the sample sizes used to estimate survey means.

Areas of sensitivity that were not tested include: the slope, intercept, number and range of data points used for H_1, the location of hypothetical points along a line (they need not be positioned as illustrated in Figure 1a), and the effect of correlation between the sampling error in survey mean X and survey mean Y. I leave these for future studies.

THE DATA

In the Eastern Townships of Quebec, Pace (1984) visited 10 lakes 5 times between May and September. One of the lakes had 3 basins, and was treated as 3 separate lakes. This gave the data set 12 sites. Triplicate measurements of TP and CHL were taken from one central location at each site. From the triplicates a single value was calculated, these single values were then compiled into a seasonal mean for each site (Table 1). Generally, sample sizes of 5 were used to estimate seasonal means, except for 3 sites where a sample size of 4 was used to estimate TP means. Pace (1984) provides details of lake chemistry and morphometry.

Pace (1984) used geometric means as data points in his

published regressions. I have used arithmetic means. They are simple and are commonly used by ecologists. This changed the regression equations only slightly:

$$LogCHL = 1.14 * LogTP - 0.60,\ r^2=0.94 \text{ for arithmetic data;}$$
$$LogCHL = 1.09 * LogTP - 0.56,\ r^2=0.93 \text{ for geometric data.}$$

Strictly speaking, because simulations are based on arithmetic means while the published regression in Pace (1984) is based on geometric means, we should be careful when drawing conclusions about his published regression from this study. We can be more confident when drawing conclusions about the TP-CHL regression that uses arithmetic means. I leave it to another to decide if such precaution is pedantry.

TABLE 1. The survey values derived from Pace (1984).

	TP			CHL		
		mean			mean	
Site	n	(mg/m^3)	S^2	n	(mg/m^3)	S^2
Bowker	4	3.75	4.8	5	1.30	0.54
Orford	4	5.33	1.5	5	1.53	0.41
Brompton	4	6.52	5.8	5	1.97	0.28
Lovering	5	7.04	1.4	5	2.50	0.18
Argent	5	10.71	5.7	5	2.45	0.29
North*	5	9.10	9.6	5	3.82	2.97
Central*	5	10.04	7.0	5	4.57	1.39
South*	5	12.79	7.0	5	5.64	2.89
Massawippi	5	12.71	8.0	5	4.19	3.54
Brome	5	14.65	8.8	5	4.07	5.90
Magog	5	47.38	44.6	5	16.88	87.60
Waterloo	5	59.70	434.7	5	34.98	387.40

* different basins in Lake Memphremagog, all other sites are separate lakes.

The non-transformed arithmetic TP-CHL data, the log-transformed data, and the hypothetical data used in simulations are shown in Figure 3.

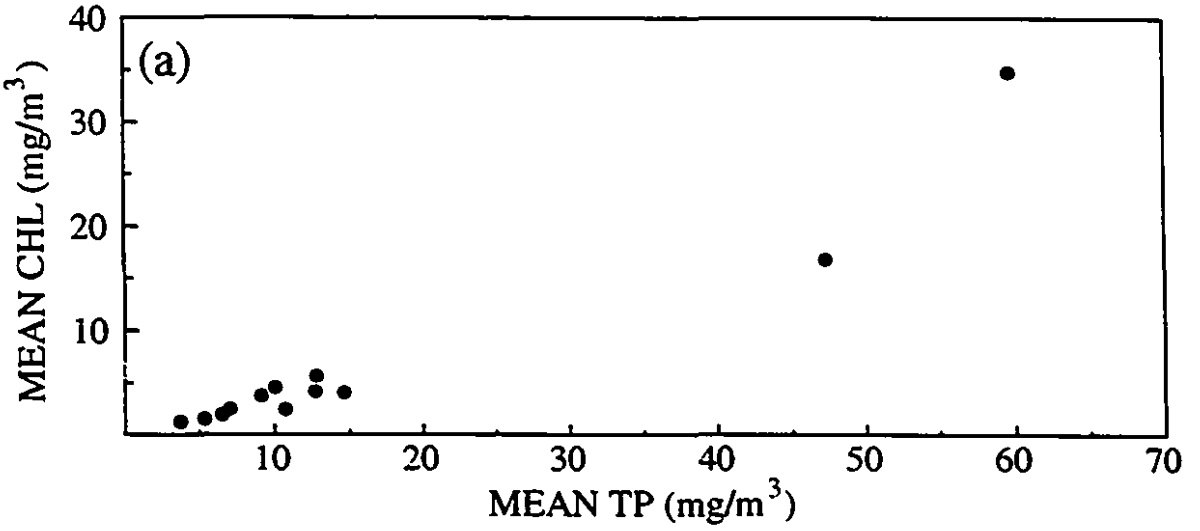
RESULTS AND DISCUSSION

PRE-SIMULATION CONCERNS

Assessing the plausibility of the hypothetical data:

In most cases the hypothetical means for both TP and CHL were deep inside the 95% confidence limits for their corresponding survey means (Figure 4). Therefore the hypothetical values appear consistent with the survey means. The use of the t-distribution can be questioned though, since the distributions of replicate TP and CHL are typically asymmetric (Heyman et al., 1984; Walmsley, 1984). This asymmetry might compromise the CLT prediction that the distributions of survey means will be Normal (Hogg and Tanis, 1983). However, the hypothetical values are generally close to the survey means and far from the 95% confidence limits; therefore small inaccuracies in estimating those confidence limits are believed to have little impact.

FIGURE 3: (a) Non-transformed seasonal mean TP and CHL data from Pace (1984). (b) Log-transformed seasonal mean TP and CHL data derived from Pace (1984). The regression equation is: LogCHL = 1.14 LogTP - 0.60, r^2 = 0.94. (c) The hypothetical data set used in simulations. Hypothetical data are placed along the regression line fit to the log-transformed survey data from Pace (1984), as shown in Figure 1.


(a)
MEAN CHL (mg/m3)
40
30
20
10
10
20
30
40
50
60
70
MEAN TP (mg/m3)

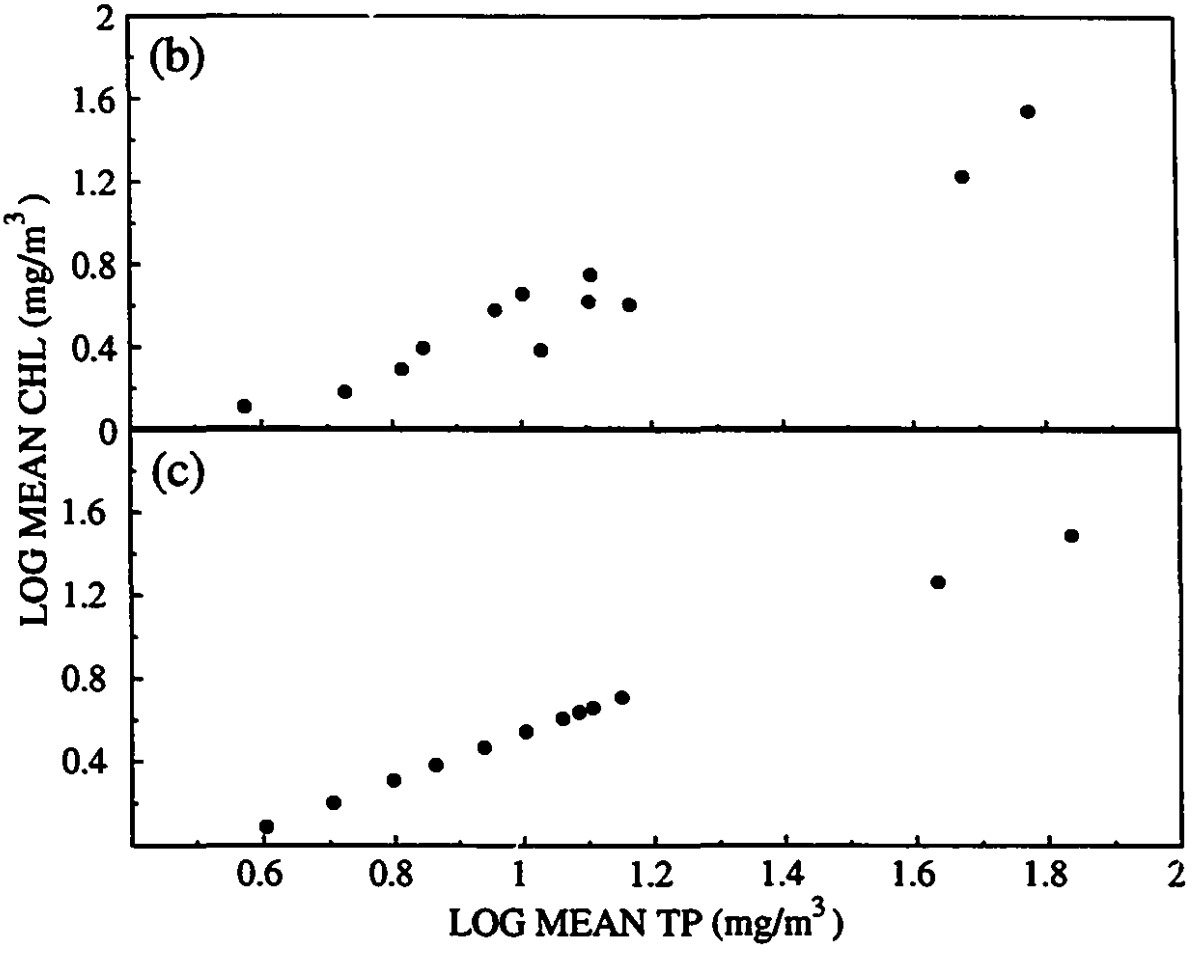
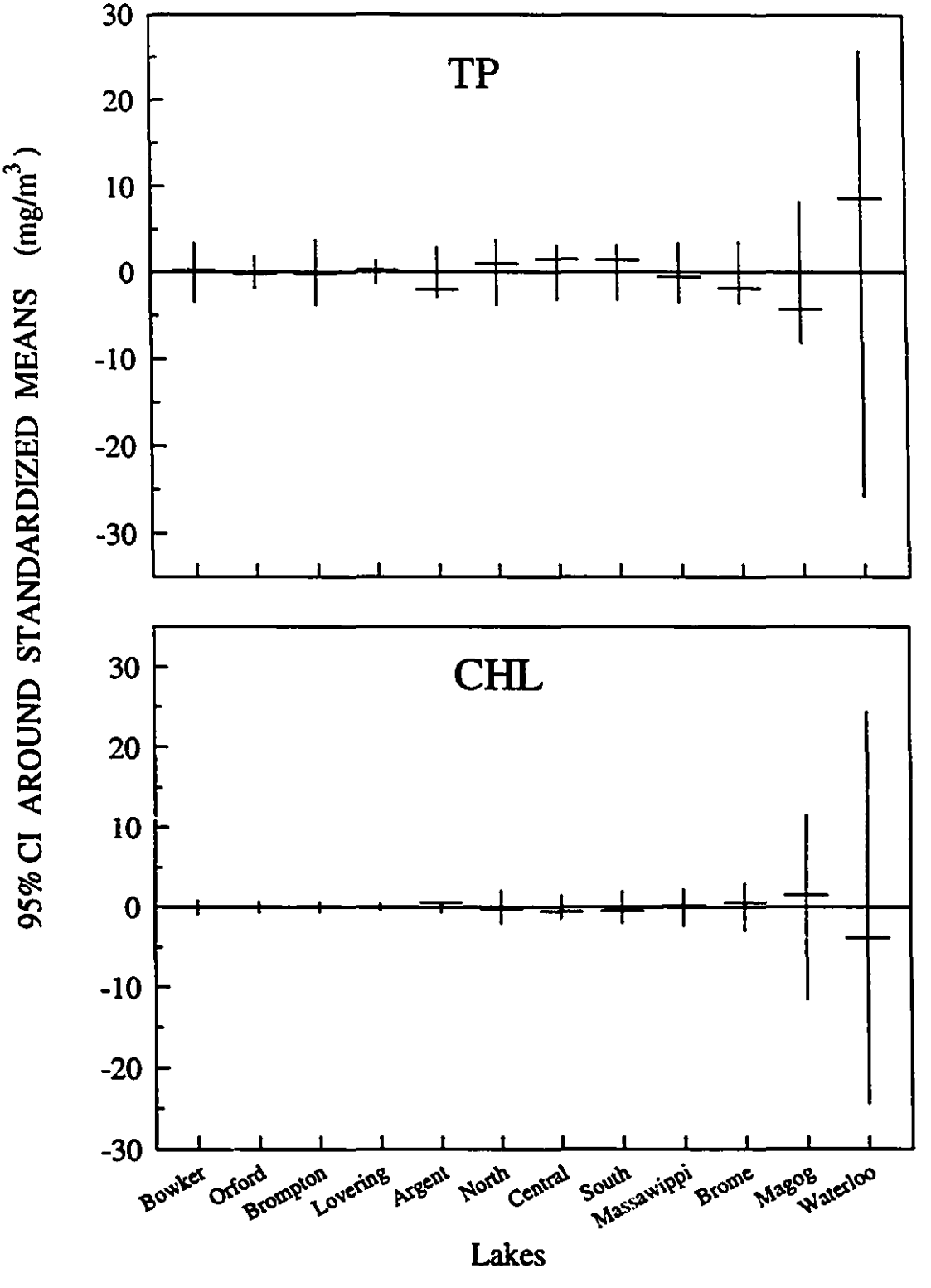

(b)
(c)
LOG MEAN CHL (mg/m3)
2
1.6
1.2
0.8
0.4
0
1.6
1.2
0.8
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
LOG MEAN TP (mg/m3)

FIGURE 4: The plausibility of each hypothetical data point depends, in part, on it being reasonably close to the survey data. Here the survey means and 95% CI for TP and CHL have been shifted so that the survey means all lie at zero. The 95% CI for the survey mean from each lake is shown by a vertical line. The hypothetical means are marked by short horizontal lines. The hypothetical means lie well within the 95% CI in all cases.


TP
CHL
95% CI AROUND STANDARDIZED MEANS (mg/m3)
30
20
10
0
-10
-20
-30
Bowker
Orford
Brompton
Lovering
Argent
North
Central
South
Massawippi
Brome
Magog
Waterloo
Lakes

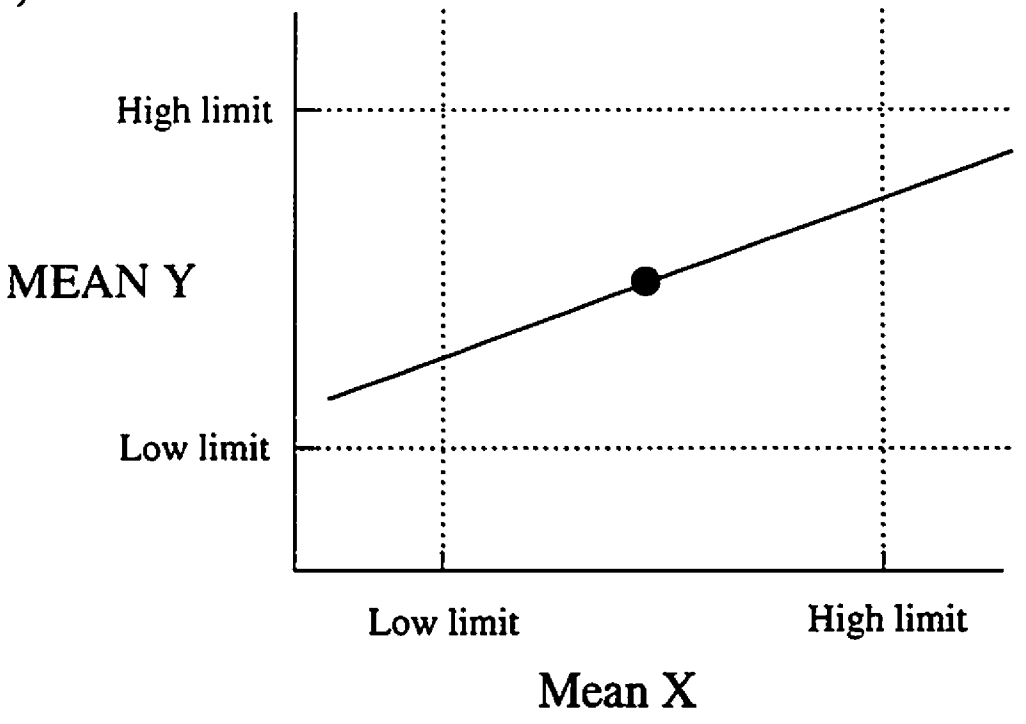
The effect of correlation between sampling error in survey mean X and survey mean Y:

The simulation program assumes that, within each lake, sampling error in a survey mean X does not covary with sampling error in survey mean Y. This assumption has been used by others, though it has dangers (Reckow, 1994). If sampling errors do covary, then the simulation results will be biased. The simulation program will tend to overestimate the effects of sampling error when the covariance is positive, and underestimate it when the covariance is negative (Figure 5).

I believe that assuming independence will not seriously bias simulation results based on the Pace (1984) data set. The double constraints imposed on simulated means (min/max and 95% CI) should counter-balance bias that might result from weak, positive correlations between sampling error in survey mean X and Y. Plots of replicate X and Y from Pace's (1984) 12 sites did not reveal any trend to strong correlation (Table 2), which is consistent with weak or no covariance between the sampling error in survey means (Figure 6). However, the sample sizes (4 or 5) used in these plots may be insufficient to expose covariance. Some larger data sets from one of the lakes that Pace surveyed did not suggest strong covariance either (Table 2).

FIGURE 5: (a) A close look at one hypothetical data point on a regression line. Dashed lines delineate the limits imposed on simulated sampling error. (b) Four arrows symbolizing sampling error divide the available space into quadrants. With a positively sloping regression line, if the sampling error in mean X and Y is positively correlated, then quadrants II and IV are more likely to be occupied, where residuals tend to be lower.

(a)


High limit
MEAN Y
Low limit
Low limit
High limit
Mean X

(b)

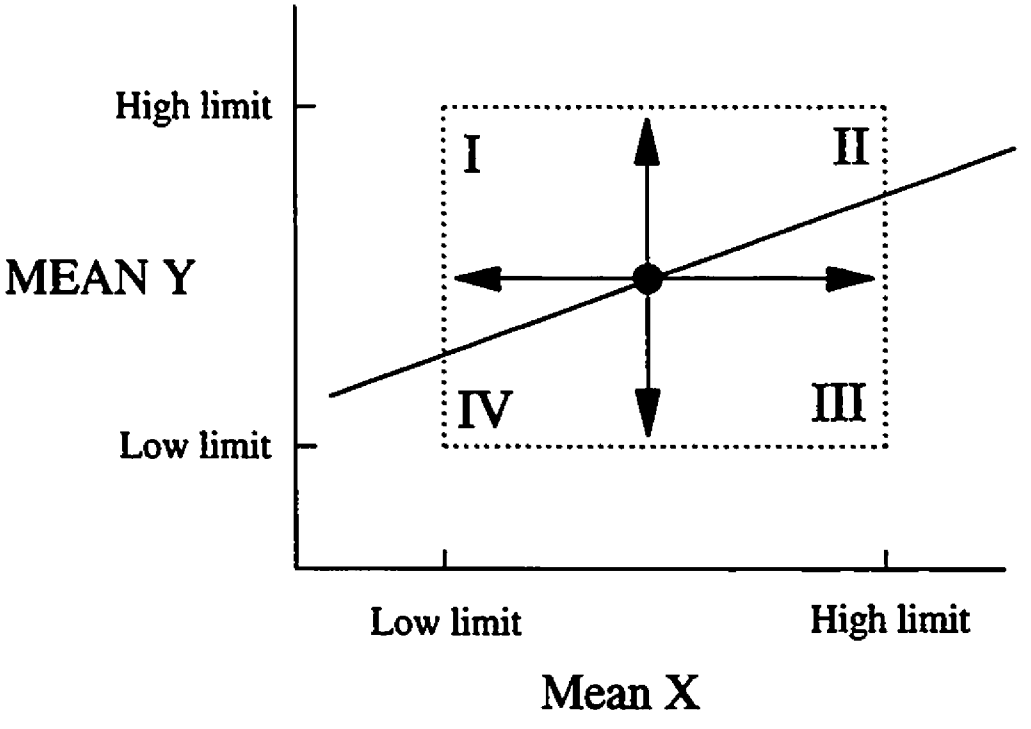
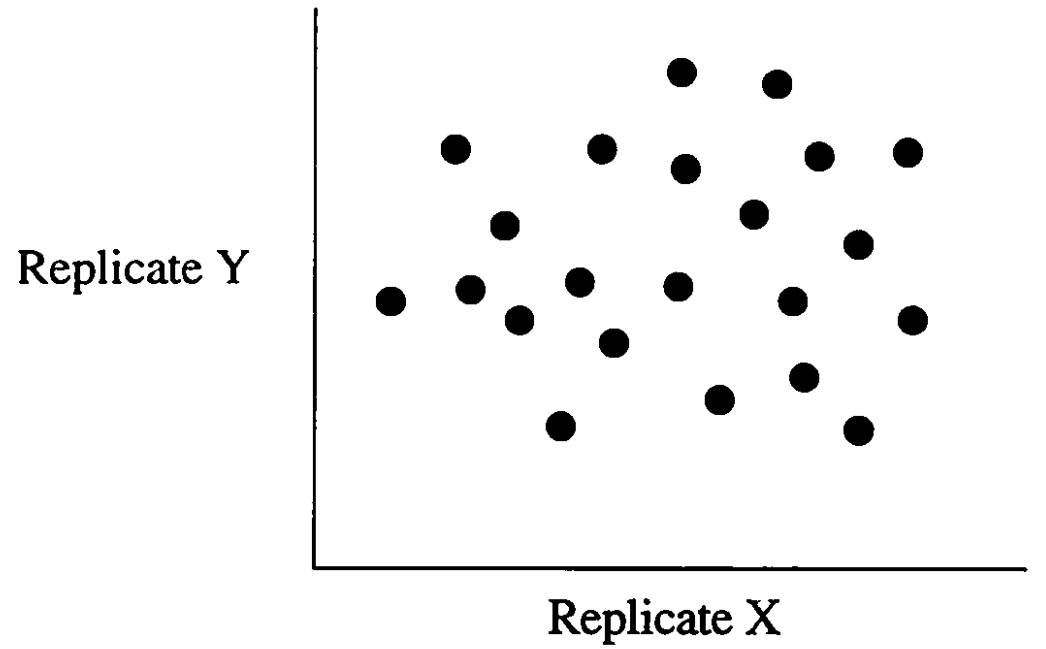

High limit
I
II
MEAN Y
IV
III
Low limit
Low limit
High limit
Mean X

FIGURE 6: Examples of systems where sampling error in mean X and mean Y are (a) independent, and (b) correlated. Correlation between replicate X and replicate Y is associated with correlation between the sampling error in Mean X and Mean Y.

(a)



(b)

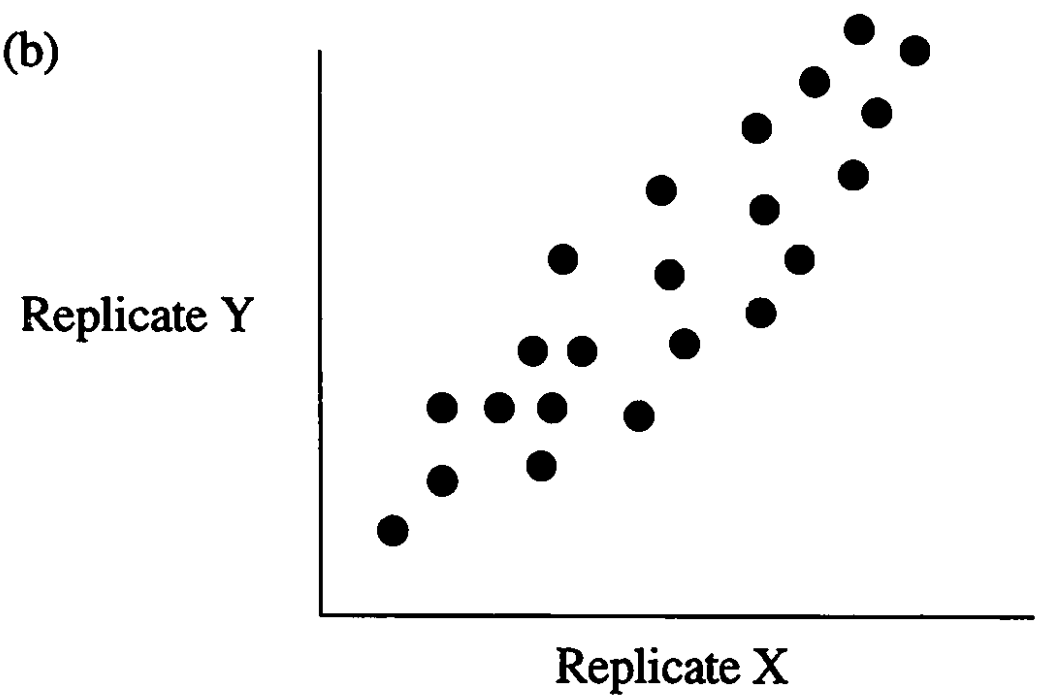


TABLE 2. Within-lake correlations between replicate TP and CHL. Data are from two sources: Pace (1984) and The McGill Limnology Research Centre. The number of replicate data points (n), r^2, and the significant r^2 at 5% level are shown. The McGill Limnology Research Centre sites are on Lake Memphremagog.

Pace data

Lake	n	r^2	significant r^2
Bowker	4	0.05	0.77
Orford	4	0.06	0.77
Brompton	4	0.97*	0.77
Lovering	5	0.23	0.66
Argent	5	0.39	0.66
North	5	0.45	0.66
Central	5	0.42	0.66
South	5	0.50	0.66
Massawippi	5	0.00	0.66
Brome	5	0.70*	0.66
Magog	5	0.02	0.66
Waterloo	5	0.49	0.66

McGill Limnology Research Centre data

Sites	Year	n	r^2	significant r^2
Pender	1975	20	0.49*	0.20
Border	1979	33	0.44*	0.13
Central	1979	33	0.09	0.12
North	1979	34	0.02	0.11
Pender	1979	33	0.37*	0.12
Border	1980	31	0.41*	0.13
Border	1981	15	0.43*	0.26
Central	1981	15	0.02	0.26
North	1981	16	0.00	0.24
Central	1985	19	0.34*	0.21

* statistically significant r^2 ($p<0.05$).

The effect of non-random sampling:

Pace (1984) sampled lakes periodically (about once a month) rather than randomly. The CLT assumes random sampling (Hogg and Tanis, 1983). If sampling is not random, then the distribution of a survey mean might differ from the CLT approximation.

Unfortunately, there is no way of knowing how much simulated distributions differ from those distributions that gave rise to Pace's (1984) survey means. This is not a unique situation. For instance, random sampling is often assumed when we make confidence intervals, even when sampling is periodic.

SIMULATION RESULTS

About 30% of the 1000 regressions on simulated data had SSE values that exceeded the SSE found using Pace's (1984) survey data (Table 3). In other words, about 30% of the survey regressions from a set of 12 lakes in which logCHL was perfectly predictable from logTP would have an SSE greater than the SSE found using Pace's (1984) data, given that survey means behave like the simulated means. Using computer generated 95% CI, the model r^2 is not significantly different than the survey r^2. The distributions of model SSE

TABLE 3. Simulation results from 1000 simulated data sets.

Probability that model SSE > survey SSE	= 30.1%
mean model r^2 (95% CI)	= 0.95 (0.89 - 0.99)
mean model SSE (95% CI)	= 0.092 (0.026 - 0.210)
<u>For comparison, derived from Pace (1984):</u>	
survey SSE	= 0.1082
survey r^2	= 0.94

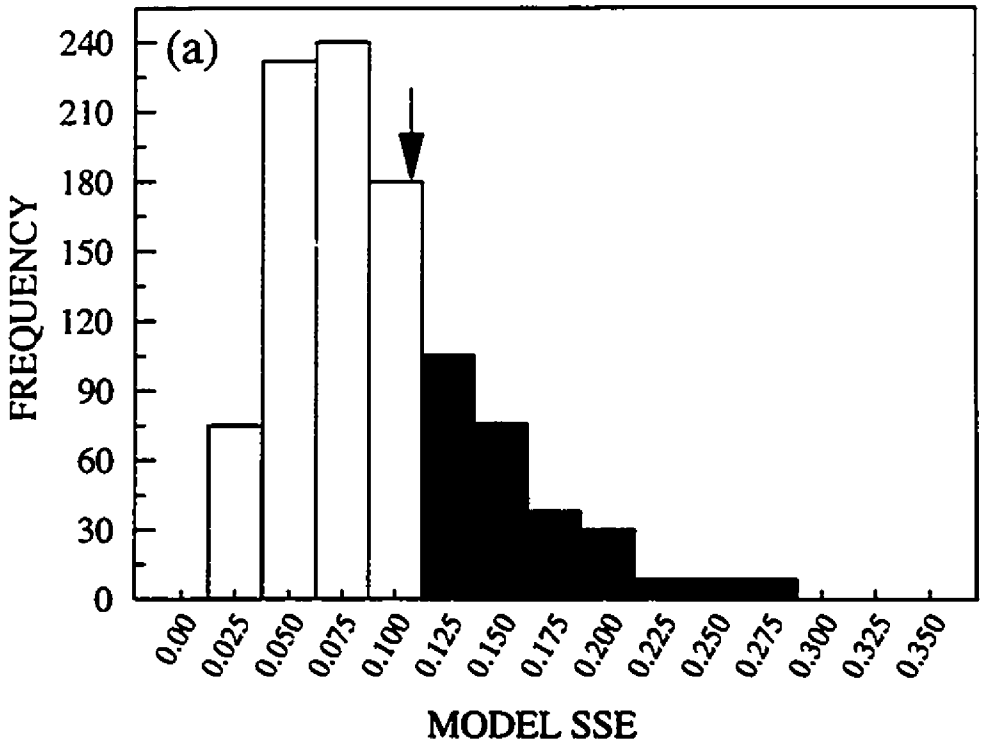
and r^2 are shown in Figure 7.

Individual simulated data sets are also consistent with the perfect-fit hypothesis. Three simulated data sets are shown in shown in Figure 8. Scatter is due only to the effects of simulated sampling. The program generated 1000 such data sets, and only three are illustrated, so there is a danger of persuasion through editing. However, the three data sets reflect the spectrum of r^2 and SSE values produced through simulation. Only Figure 8a has an SSE greater then the survey SSE. This corresponds roughly to the 30% of all simulated data sets that had higher SSE values than the survey SSE.

Though it is counter-intuitive to believe the perfect-fit hypothesis (it seems unlikely that CHL could be entirely predicted by TP), the possibility that most of the variation in CHL could be explained by TP cannot be dismissed out of hand. This is disturbing. By exposing such ambiguity in one regression analysis, the simulation results force us to question all interpretations that do not consider the "perfect-fit" null hypothesis.

These results evoke a second question: when faced with residuals in a Y on X regression, is it better to diffuse our effort searching for other predictors (X_2,X_3,X_4, and so on ...), or to focus effort on achieving better estimates of X and Y? (Of course, it would be nice to do both).

FIGURE 7: (a) The distribution of model SSE, from 1000 simulated data sets. The SSE found using Pace's (1984) data was 0.1082 and is shown by a vertical arrow. (b) The distribution of model r^2, from the same 1000 simulated data sets. The r^2 found using Pace's (1984) data was 0.94 and is shown by a vertical arrow. Shading emphasizes where model SSE exceeds the survey SSE, or model r^2 is less than the survey r^2.


(a)
FREQUENCY
240
210
180
150
120
90
60
30
0
0.00
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275
0.300
0.325
0.350
MODEL SSE

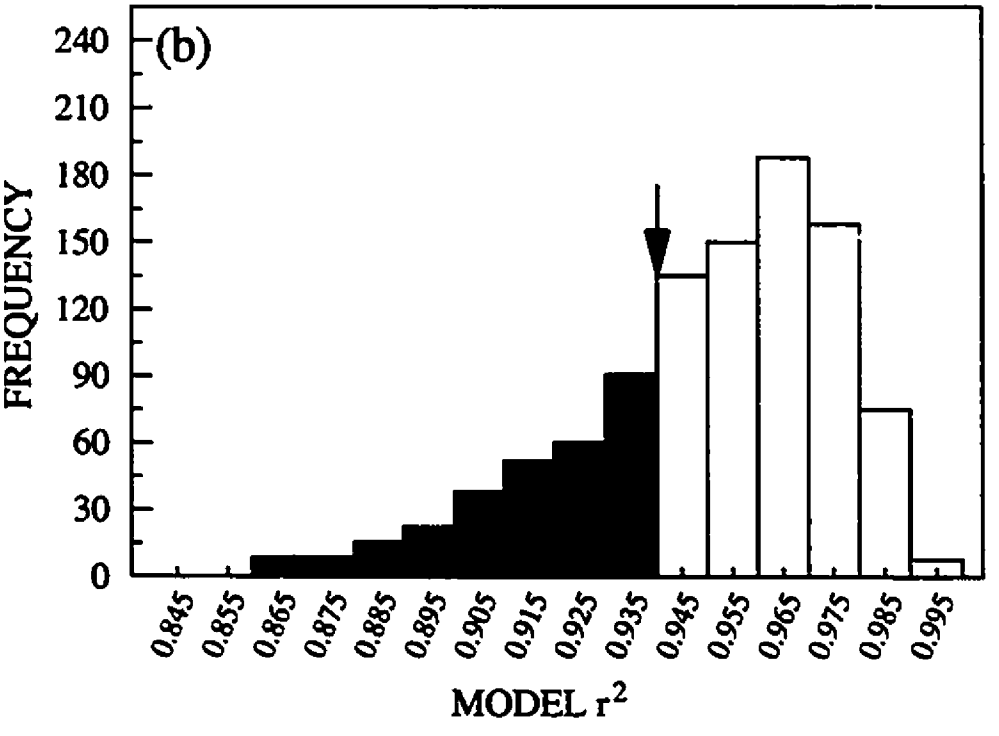
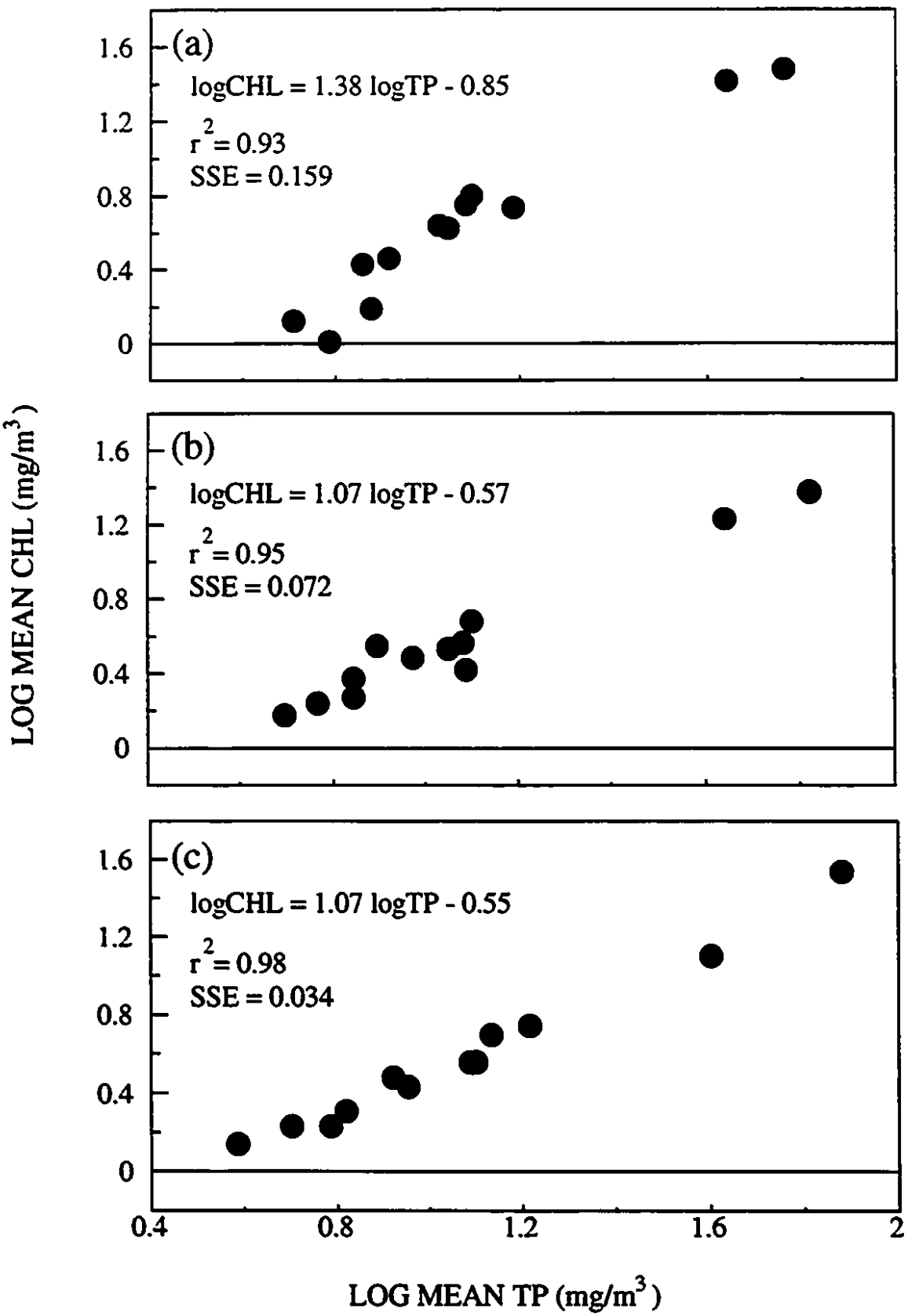

(b)
FREQUENCY
240
210
180
150
120
90
60
30
0
0.845
0.855
0.865
0.875
0.885
0.895
0.905
0.915
0.925
0.935
0.945
0.955
0.965
0.975
0.985
0.995
MODEL r2

FIGURE 8: Three simulated data sets were chosen to provide examples of SSE greater than, about the same as, and less than the survey SSE (0.1082). (a) The first simulated data set. (b) The third simulated data set. (c) The fifth simulated data set.


(a)
logCHL = 1.38 logTP - 0.85
r² = 0.93
SSE = 0.159
(b)
logCHL = 1.07 logTP - 0.57
r² = 0.95
SSE = 0.072
(c)
logCHL = 1.07 logTP - 0.55
r² = 0.98
SSE = 0.034
1.6
1.2
0.8
0.4
0
0.4
0.8
1.2
1.6
2
LOG MEAN CHL (mg/m³)
LOG MEAN TP (mg/m³)

This study implies that achieving better estimates of mean TP and CHL could be more informative than sampling additional variables at the same intensity as the initial study.

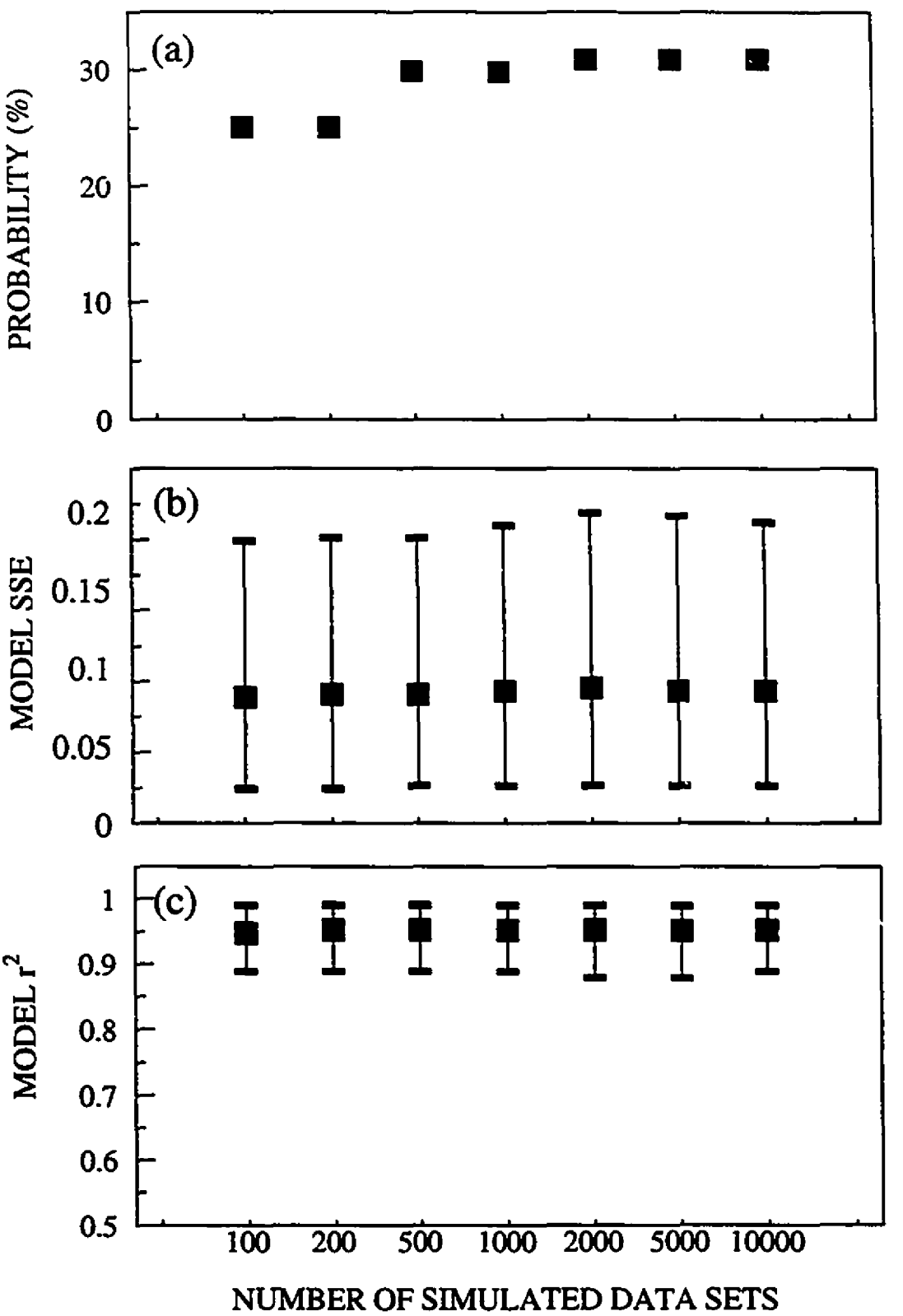
SENSITIVITY TESTS

Three criteria are used to assess sensitivities: the probability that model SSE exceeds survey SSE, mean model SSE and its 95% CI, and mean model r^2 and its 95% CI.

The effect of altering the number of simulated data sets:

Analyses were done using 100, 200, 500, 1000, 2000, 5000 and 10000 simulated data sets. A sample size of 500 was sufficient to produce stable results for the probability that model SSE exceeded survey SSE (Figure 9a). All sample sizes produced similar mean model SSE values (Figure 9b). Likewise, all sample sizes produced similar mean r^2 values (Figure 9c). Considering these findings, a sample size of 500 may have been adequate, but a sample size of 1000 seemed to offer a better blend of computing time and credibility.

FIGURE 9: Results using different numbers of simulated data sets. (a) The probability that model SSE exceeds survey SSE. (b) The mean and 95% CI for model SSE. (c) The mean and 95% CI for model r^2.


(a)
PROBABILITY (%)
30
20
10
0
(b)
MODEL SSE
0.2
0.15
0.1
0.05
0
(c)
MODEL r^2
1
0.9
0.8
0.7
0.6
0.5
100
200
500
1000
2000
5000
10000
NUMBER OF SIMULATED DATA SETS

The effect of using non-CLT dependent simulations:

The result that 30% of model SSE exceeded survey SSE came from simulations based on the CLT. The CLT generally gives good approximations if n is greater than 25 or 30, where n is the sample size used to calculate the sample mean. If the underlying distribution (the distribution of replicate values) is symmetric, unimodal, and continuous, then good approximations can be had when n is as low as 4 or 5 (Hogg and Tanis, 1983). Pace (1984) used sample sizes of 4 or 5, but the underlying distributions of his survey mean TP and CHL were possibly logNormal (Pace, 1984), and therefore asymmetric. If so, the sample sizes provided by Pace (1984) may have been too small for the CLT to give good approximations of the distributions of means.

To investigate this problem, a second simulation program was made that did not use the CLT. The original simulation program (CLT program) chooses means directly from a distribution of means approximated by the CLT. The second program instead calculates each mean from n simulated replicates (n is the number of replicates used by Pace (1984) to estimate each survey mean).

The second simulation program (logNormal replicate program) assumes that all underlying distributions are logNormal. The means of these underlying distributions are the same hypothetical means used in the CLT program, and the

variances are the S^2 values measured by Pace (1984). The computer code appears in Appendix II.

In the CLT program the following equation is used to calculate 95% CI for simulated means:

95% confidence limit= hypothetical mean +/- 1.96 * $(S^2/n)^{1/2}$

But this equation is appropriate only when the CLT applies, so it is not used in the logNormal replicate program. Instead, 95% CI are based on simulations. For each of the hypothetical mean TP and CHL values, 1000 simulated means are made by averaging n simulated replicates, 1000 times. Each collection of 1000 simulated means is ordered from lowest to highest value; the 25th and 975th values are used as the 95% confidence limits.

A third simulation program (Normal replicate program) was also made. Its broad structure is identical to the logNormal replicate program, except that the Normal replicate program assumes that all underlying distributions are Normally distributed.

This third program acts as a control in comparisons between the simulation results of the logNormal replicate program and CLT program. If results from the CLT program are similar to results from the Normal replicate program, then any difference between results from the CLT program and the logNormal replicate program must stem from their different

assumptions about the underlying distributions (Normal versus logNormal). Without the control, results would be more ambiguous, because one would not be able to rule out the possibility that different results might stem from the different ways that the CLT program and logNormal replicate program simulate means: the CLT program simulates means directly while the logNormal replicate program first simulates replicates.

Both the CLT program and Normal replicate program show that the probability of model SSE exceeding survey SSE is about 30% (Table 4). The logNormal replicate program gives a probability that is about 5% less. For model SSE and r^2, all programs gave similar mean values, but differed somewhat in 95% CI. The conclusion, therefore, is that we can have some confidence that we can approximate the effects of sampling error on a regression, despite our ignorance about the shape of underlying distributions. Complete confidence is not warranted, however, because underlying distributions may be neither Normal nor logNormal.

Only the CLT program is used in the rest of the study. It is referred to, as before, as "the simulation program".

TABLE 4. Results using three different assumptions about the shapes of the distributions of replicate TP and CHL: the CLT assumptions; all distributions of replicate TP and CHL are logNormal, the CLT is not applied; all distributions of replicate TP and CHL are Normal, the CLT is not applied. Shown are the probability that model SSE exceeds survey SSE ("P"), the mean of model SSE and its 95% CI, and the mean of model r^2 and its 95% CI. Apart from this sensitivity test, the CLT assumptions were used throughout this study.

Assumptions	P	Model SSE (95% CI)	Model r^2 (95% CI)
CLT	30	0.092 (0.026 - 0.210)	0.95 (0.89 - 0.99)
LogNormal replicates	26	0.088 (0.027 - 0.188)	0.95 (0.89 - 0.99)
Normal replicates	31	0.094 (0.027 - 0.227)	0.95 (0.88 - 0.98)

The effect of altering min and max used as bounds in simulation:

Two additional sets of simulations were done using different min/max values. One set of simulations ("narrow") used min/max that were close to the hypothetical means, another set ("wide") used min/max that were far from hypothetical means.

In each lake, the "narrow" min was calculated by halving the distance between the hypothetical mean and the lowest sampled replicate value. The "narrow" max was calculated by halving the distance between the hypothetical mean and the highest sampled replicate value. The "wide" min was calculated by halving the lowest sampled replicate value, the max by multiplying the highest replicate value by 1.5.

Sometimes when "narrow" min/max are used, limits to simulated values are exceeded by hypothetical means themselves (before there is any simulation). In these situations, either the min/max limits or the hypothetical means are inappropriate. Hypothetical means exceeded "narrow" min/max limits 4 times out of the 24 (17%) using the Pace (1984) data. Since these min/max limits were chosen to test the effects that the min/max limits have on simulations, and were not an attempt to represent real min/max limits, it may not matter whether the hypothesis or

min/max limit was inappropriate in four cases. To avoid faulty simulations, limits were widened, in those four cases, to include the hypothetical mean.

Under "narrow" min/max limits, the probability of model SSE exceeding survey SSE was 9% (Table 5). Under the regular min/max limits, the probability was 30%. Under "wide" min/max limits, the probability was 34%. Therefore, how we choose min/max limits certainly affects these probabilities.

Simulations predict a 3% to 5% reduction in r^2, on average, due to sampling error, regardless of the min/max limits used (Table 5). The average model SSE and r^2 are more robust with respect to min/max limits than is the probability of model SSE exceeding survey SSE. There is almost a fourfold difference between the "narrow" and "wide" results for the probability of model SSE exceeding survey SSE, which reveals this question's requirement for accurately estimated min/max. How to best estimate min/max is unresolved.

The effect of altering S^2 estimates:

Each simulated mean comes from a distribution that is Normal and has a variance of S^2/n, where S^2 is the variance of replicates surveyed by Pace (1984) and n is the sample size he used to estimate means. The amount of sampling error

TABLE 5. Results using different criteria to set min/max for simulated mean TP and CHL. "Narrow", "regular", and "wide" are described in text. Shown are the probability that model SSE exceeds survey SSE ("P"), the mean of model SSE and its 95% CI, and the mean of model r^2 and its 95% CI.

Min/max	P	Model SSE (95% CI)	Model r^2 (95% CI)
Narrow	9	0.065 (0.019 - 0.138)	0.97 (0.92 - 0.99)
Regular	30	0.092 (0.026 - 0.210)	0.95 (0.89 - 0.99)
Wide	34	0.099 (0.028 - 0.230)	0.95 (0.88 - 0.98)

allotted to each hypothetical mean during simulation depends on S^2/n, which in turn affects model SSE. This section discusses S^2 effects, the next section discusses n effects.

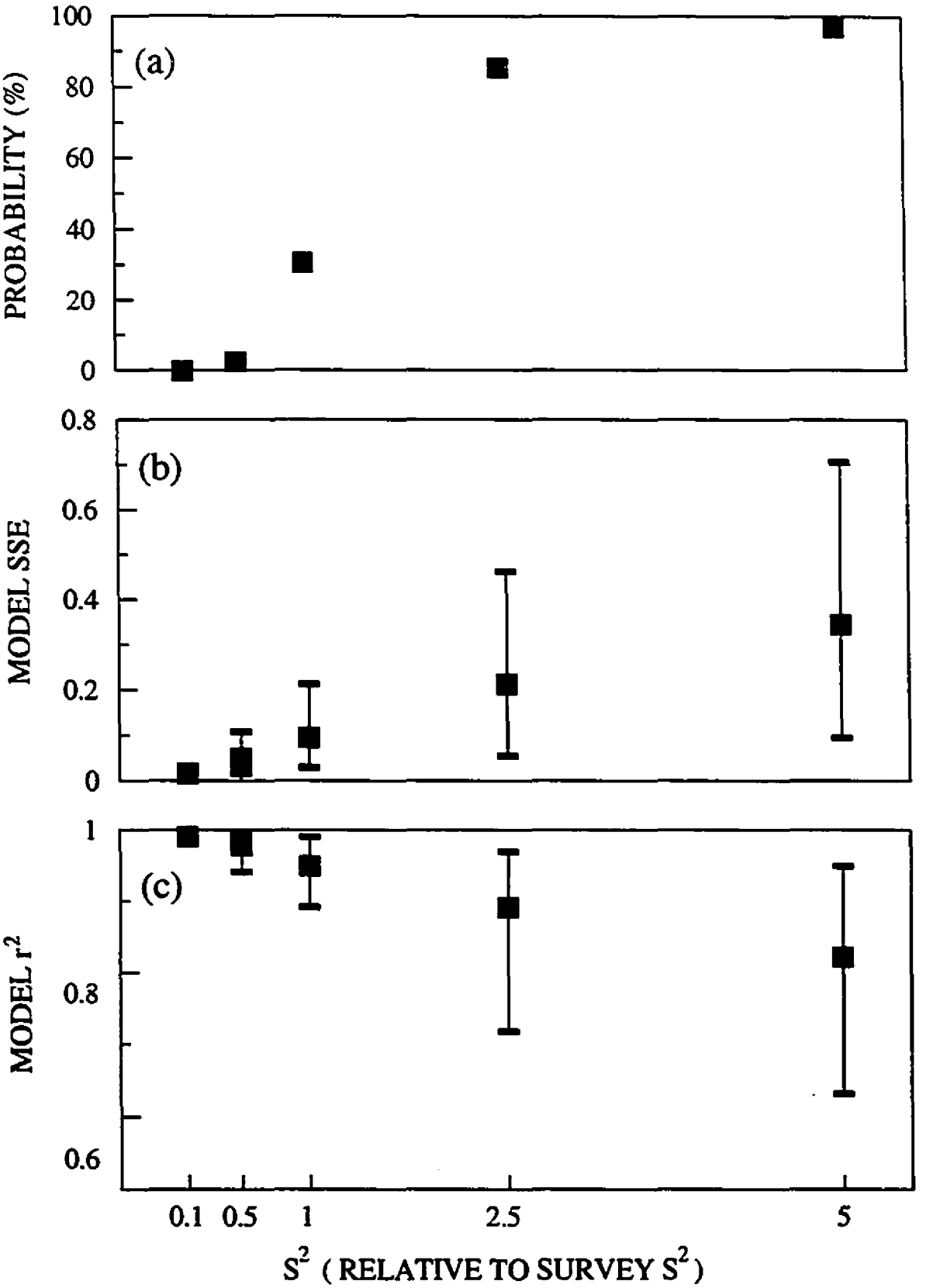
Simulations used four sets of S^2 values, in addition to the S^2 values from sampled replicates (called regular S^2). Each of the four sets was made by multiplying each of the 24 regular S^2 values by a factor of: 0.1, 0.5, 2.5, or 5. One way to give these factors a context is to look at confidence intervals for S^2. Using the method of shortest unbiased confidence intervals (Sokal and Rohlf, 1981), when sample size is five, the 95% confidence interval for the S^2 of a normal distribution is:

$$0.3125 * S^2 \quad \text{to} \quad 6.590 * S^2 \quad .$$

When S^2 values are set at 1/10th the S^2 of sampled replicates, the probability of model SSE exceeding survey SSE is zero. When S^2 values are set at five times the S^2 of sampled replicates, the probability is 97% (Figure 10). Mean model SSE and r^2 also show wide spreads due to changes in S^2 (Figure 10).

These findings may not be as troublesome as they first appear. When the S^2 factors shown in Figure 10 were applied, the same factor was applied to all of the 24 underlying distributions for TP and CHL, and then 1000 data sets were simulated. Then a new S^2 factor was applied and the process

FIGURE 10: Results assuming different S^2 estimates for replicate TP and CHL. (a) The probability that model SSE exceeds a fixed SSE (0.1082, the survey SSE from Pace (1984)). The range in S^2 values was achieved by multiplying Pace's S^2 measurements by 0.1, 0.5, 1, 2.5 and 5. (b) The mean and 95% CI for model SSE. (c) The mean and 95% CI for model r^2.

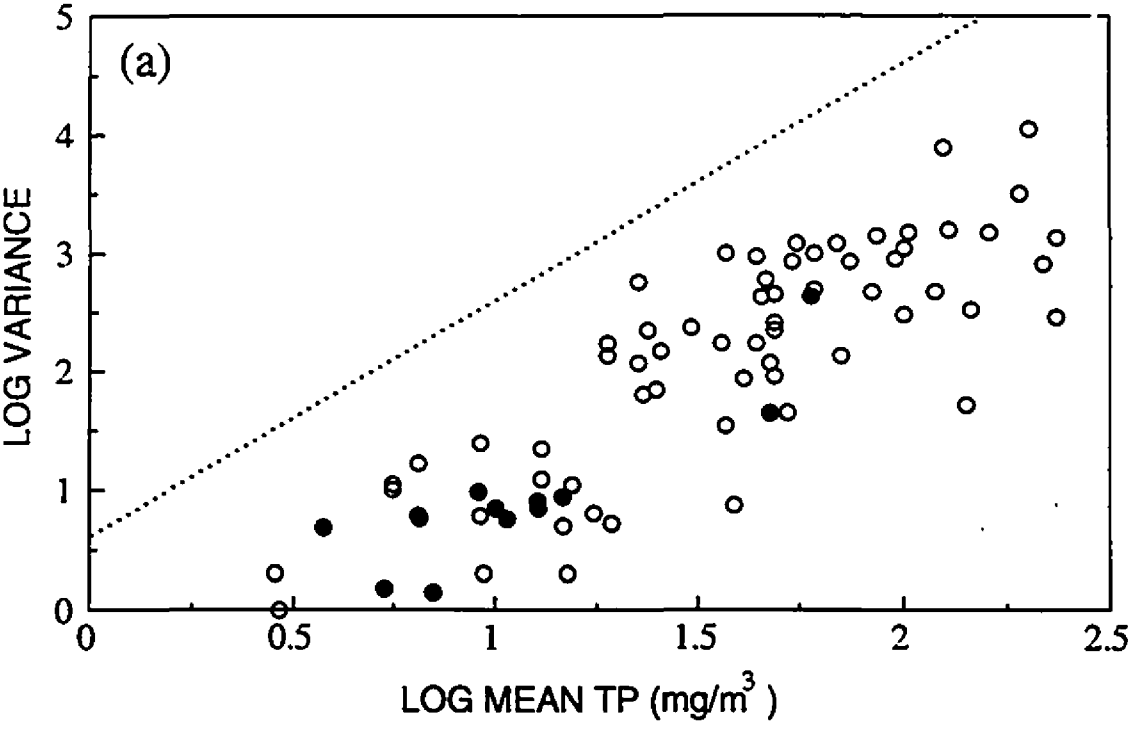

(a)
PROBABILITY (%)
100
80
60
40
20
0
(b)
MODEL SSE
0.8
0.6
0.4
0.2
0
(c)
MODEL r²
1
0.8
0.6
0.1
0.5
1
2.5
5
S² (RELATIVE TO SURVEY S²)

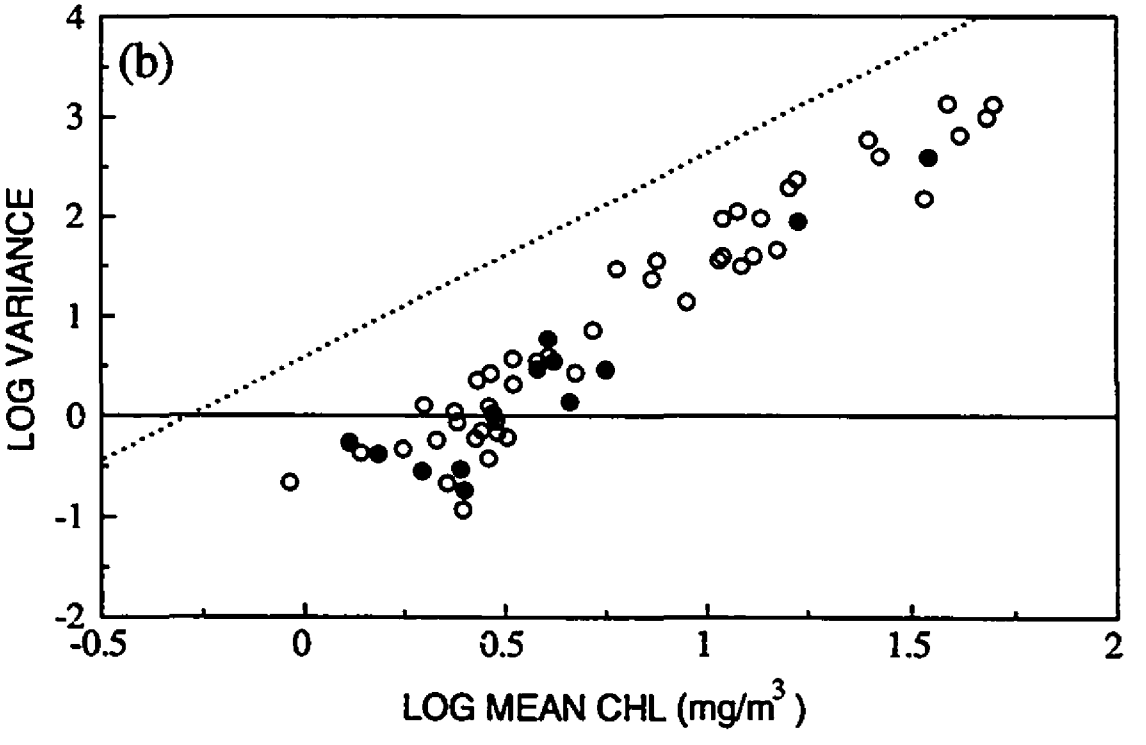
repeated. But in a well designed study, the chance of every S^2 estimate being very low, or very high, relative to the true values is minute. For instance, a factor of 0.3 falls at the lower edge of the 95% confidence interval for an S^2 from a sample size of five from a Normal distribution. The chance of every sampled S^2 value being that low is about $4 * 10^{-39}$, and would on average require $2.5 * 10^{38}$ surveys before happening.

Because one would expect overestimates of S^2 as often as underestimates of S^2, the effects of inaccurate S^2 estimation may not be large. In fact, this seems to be the case with Pace's (1984) S^2 measurements. When his S^2 measurements are plotted on published mean-variance regressions for TP (France and Peters, 1992) and CHL (Marshal et al., 1988) they are not consistently high; some are high, some are low, some are in the middle (Figure 11).

S^2 has an upperbound of nM^2 (where n is the sample size used to estimate the mean, M) (Appendix I). It is possible that Pace's (1984) S^2 values are highly inflated but fall within the mean-variance plots in Figure 11 because his values are more mathematically constrained than those values in the published plots (France and Peters (1992), for instance, use n=7). The dashed line in Figure 11 shows nM^2 for n=5, the sample size most used by Pace (1984). Pace's (1984) S^2 values are about 1/10th to 1/100th of the nM^2 maximum, therefore there is no evidence of mathematical

FIGURE 11: Mean:variance plots can be used to show that Pace's (1984) S^2 measurements are not unusual. (a) Pace's measurements of S^2 associated with TP (solid circles) superimposed on a mean:variance plot for TP from France and Peters (1992) (open circles). (b) Pace's measurements of S^2 associated with CHL (solid circles) superimposed on a mean:variance plot for CHL from Marshall et al. (1988) (open circles). The dotted lines show the theoretical nM^2 maximum for S^2 for Pace's results (n=5), and that mathematical constraints are not the reason that Pace's results fall within those of the general mean:variance plots.


(a)
LOG VARIANCE
LOG MEAN TP (mg/m3)
0
1
2
3
4
5
0
0.5
1
1.5
2
2.5


(b)
LOG VARIANCE
LOG MEAN CHL (mg/m3)
-2
-1
0
1
2
3
4
-0.5
0
0.5
1
1.5
2

constraint at a given level of M. Another criticism might be that the published mean-variance plots provide an "easy" test if some of their S^2 values are inflated due to low sample size. The France and Peters (1992) plot, however, used n=7 to calculate each of their data points (Figure 11a), which is larger than any sample size used by Pace (1984). The sample size criticism may apply to the Marshall et al. (1988) plot (Figure 11b); but if it does, then it also suggests that some of the S^2 values are underestimated in this plot, which would suggest that any of Pace's S^2 values that fall along the bottom of this plot are especially low. A reasonable conclusion is that Pace's values do not appear unusual. This study's findings, therefore, may be applicable to TP-CHL relationships in general.

The effect of altering the sample size used to estimate the mean:

Because sample size is almost always known without error, it does not affect the robustness of the simulation results. But the effects of sample size are interesting since sample size is something which researchers control.

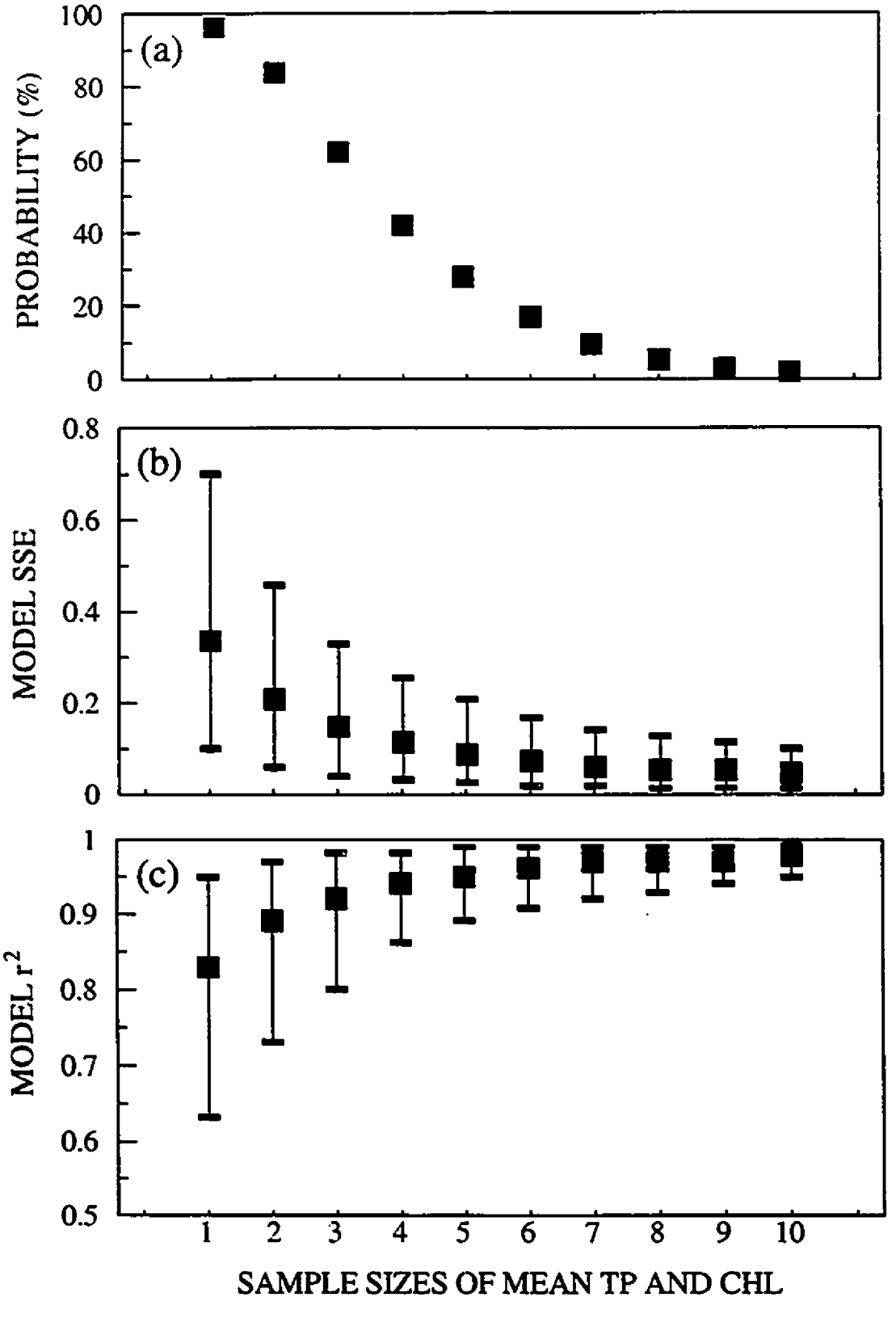
The simulation program was run with all n = 1, then with all n = 2, then 3, and so on until all n = 10. The

probability that model SSE exceeded survey SSE falls from 96% to 2% as sample size increases from one to ten (Figure 12a). As n increases, the effects of sampling error become increasingly difficult to reduce by further adjustments to sample size. This pattern should be quite general, since it stems from the fact that sample sizes can be incremented at a constant rate to a practically limitless number, whereas the probability has a lower bound of zero.

Graphs such as these could help researchers plan sampling programs. For instance, if we want to reduce the probability of model SSE exceeding some fixed SSE (0.1082 in Figure 12a) from 30% to less than 5%, a sample size of nine is needed. However, after a sample size of about 6 or 7, gains in the precision of SSE estimates become very small.

There is a growing body of literature that uses mean-variance relationships to predict the sample sizes required to estimate mean values with desired precision. Examples include sample size predictions for TP (France and Peters, 1991), CHL (Marshall et al., 1988), zooplankton (Downing et al., 1987), stream benthos (Morin, 1985), lake and river benthos (Downing, 1979), aquatic macrophytes (Downing and Anderson, 1985), and epiphytic invertebrates (Downing and Cyr, 1985). However, none of these studies looks at the effect of precision on regression analysis. Knowlton et al. (1984) stressed that sampling intensity must satisfy the requirements of whichever statistical test is used.

FIGURE 12: Results when the sample sizes used to calculate the standard error of the hypothetical means are altered. S^2 values are those measured by Pace (1984), and are kept constant throughout. (a) The probability that model SSE exceeds a fixed SSE (0.1082, the survey SSE from Pace (1984)). (b) The mean and 95% CI for model SSE. (c) The mean and 95% CI for model r^2.


(a)
PROBABILITY (%)
100
80
60
40
20
0
(b)
MODEL SSE
0.8
0.6
0.4
0.2
0
(c)
MODEL r²
1
0.9
0.8
0.7
0.6
0.5
1
2
3
4
5
6
7
8
9
10
SAMPLE SIZES OF MEAN TP AND CHL

Researchers who intend to use their measurements for regression analysis should not assume that sampling error will be unimportant if they follow the recommendations in the mean-variance literature. This is not a criticism of these interesting papers; it uses those ideas as a platform for new ones. France and Peters (1991), for instance, recommend that sampling programs designed to develop TP-CHL relationships should use five to seven temporal replicates. This study shows that sampling error can still be a problem when five replicates are used.

Figure 12b is based on the same model SSE distributions as Figure 12a, and the information in Figure 12a can be approximated by the information in Figure 12b. For instance, in Figure 12b the lower 95% confidence limit for model SSE is about 0.1 when sample size is one. This is close to the survey SSE in Pace's (1984) study. Therefore, one would expect that the probability of model SSE exceeding the survey SSE in Pace's (1984) study would be about 98%, when sample size is set at one. This is what Figure 12a shows.

It is sometimes difficult to assess the importance of "raw" SSE values (i.e. what does an SSE of 0.1082 mean?). Figure 12c shows sample size comparisons in terms of r^2. Keeping in mind that results are based on one data set and all of the assumptions in the simulation program, it can be interpreted as follows. When we sample a system whose mean X and Y values fall perfectly along a line, the resultant

regression is likely to show an r^2 less than one. How much less will be determined, in part, by the sample sizes used to estimate the mean values.

Under the program assumptions, Figure 12c shows that 0.80 is the lower 95% confidence limit for model r^2 from perfectly correlated hypothetical means that are based on a sample size of three. Yet, given a regression with an r^2 of 0.80, few scientists would recognize that they might be dealing with a perfect-fit. In fact, if S^2 values were high enough, and the range of X small enough, one could lose all of a perfect relationship to the effects of sampling error.

When sampling error is not large enough to obscure the relationship between a first predictor variable and Y, it can still diminish the statistical significance of a second predictor variable. This should be a concern especially when the first predictor accounts for much of the variation in Y, because then it is easier for sampling error to overshadow any additional variation in Y attributable to the second variable. This has great relevance to sampling design, and might be particularly relevant to "old questions", such as nutrient-CHL regressions, where existing models already explain much of the variation in Y and small increments in predictive power are sought through yet another variable. The interpretations of previous negative findings in those sorts of studies should be reconsidered.

CONCLUSIONS

This study investigates the hypothesis that a set of true mean TP and CHL values is transposed, through sampling error, from being perfectly correlated into the data set measured by Pace (1984). A simulation program was developed to help answer this question, a set of simple sensitivity tests assessed the applicability of simulations to Pace's (1984) work, and the robustness of the simulation program's results.

Two specific questions were asked: (1) How often do regressions through simulated data result in an SSE that exceeds the survey SSE? (2) Is the average r^2 from regressions through simulated data significantly different than the survey r^2?

The answers are: (1) Regressions through simulated data result in SSE that exceeds survey SSE about 30% of the time. This might be called a "cautious overestimate". It is cautious because simulated values were constrained, and it may be an overestimate because within each lake sampling error in mean X was assumed to be independent of sampling error in mean Y. Both aspects of the simulation were reasonable, however, and I do not think that either seriously biased the results. Instead, my confidence in the simulations is increased when I consider their counter-balancing effects. (2) The mean r^2 of regressions through

simulated data was not significantly different than the survey r^2. The criteria for significance were 95% confidence limits derived from 1000 regression analyses through 1000 simulated data sets.

These results are consistent with the hypothesis that sampling error in mean TP and CHL accounts for all of the residual error in the regression through Pace's (1984) TP-CHL data. We should therefore question whether looking for additional predictor variables is the best way to improve predictions of mean CHL in these lakes. Instead, improving the precision of the estimates of means may be more effective.

A central assertion of this study is that when mean values are used in a regression analysis, then dependent and independent variables are measured with error. This assertion conflicts with an important assumption underlying least squares linear regression. By investigating the consequences of these conflicts, we enhance our ability to interpret regression results. The simulation approach used in this study should extend beyond limnology; it should be applicable to any least squares regression study involving sample means.

REFERENCES

Dillon, P.J., and F.H. Rigler. 1974. The phosphorus-chlorophyll relationship in lakes. Limnol. Oceanogr. 19: 767-773.

Downing, J.A. 1979. Aggregation, transformation, and the design of benthos sampling programs. J.Fish.Res.Board Can. 36:1454- 1463.

Downing, J.A., and M.R. Anderson, 1985. Estimating the standing biomass of aquatic macrophytes. Can.J.Fish.Aquat.Sci. 42:1860-1869.

Downing, J.A., and H. Cyr, 1985. Quantitative estimation of epiphytic invertebrate populations. Can.J.Fish.Aquat.Sci. 42:1570-1579.

Downing, J.A., M. Perusse, and Y. Frenette. 1987. Effect of interreplicate variance on zooplankton sampling design and data analysis. Limnol. Oceanogr. 32:673-680.

Canfield, D.E.Jr., J.V. Shireman, D.E. Colle, W.T. Haller, C.E. Watkins, II, and M.J. Maceina. 1983. Prediction of chlorophyll a concentrations in Florida Lakes: importance of aquatic macropytes. Can.J.Fish.Aquat.Sci. 41:497-501.

Cooper, D., and M. Clancy, 1985. Oh! Pascal! 2nd ed. W.W.Norton & Company, New York.

France, R.L., and R.H. Peters, 1992. Temporal variance function for total phosphorus concentration. Can.J.Fish.Aquat.Sci. 49:975-977.

France, R.L., R.H. Peters, and Y.T. Prairie. 1994. Adjusting chlorophyll-a estimates through temporal weighting based on the seasonal development of phytobiomass. Aquatic Sciences 56:106-114.

Heyman, U., S. Ryding, and C. Forsberg. 1984. Frequency distributions of water quality variables. Relationships between mean and maximum values. Wat. Res. 18:787-794.

Hogg, R.V., and E.A. Tanis, 1983. Probability and Statistical Inference. 2nd ed. Macmillan Publishing Co.,Inc., New York.

Johnson, R.A., and D.W. Wichern, 1982. Applied Multivariate Statistical Analysis. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Knowlton, M.F., M.V. Hoyer, and J.R. Jones. 1984. Sources of variability in phosphorus and chlorophyll and their effects on use of lake survey data. Wat. Res. Bull. 20:397-407.

Marshall, C.T., A. Morin, and R.H. Peters, 1988. Estimates of Mean Chlorophyll-a Concentration: Precision, Accuracy, and Sampling Design. Water Resources Bulletin 24:1027-1034.

McQueen, D.J., J.R. Post, and E.L. Mills. 1986. Trophic relationships in freshwater pelagic ecosystems. Can.J.Fish.Aquat.Sci. 43:1571-1581.

Molot, L.W., and P.J. Dillon. 1991. Nitrogen/phosphorus ratios and the prediction of chlorophyll in phosphorus-limited lakes in central Ontario. Can.J.Fish.Aquat.Sci. 48:140-145.

Morales-Baquero, R., J.M. Conde-Porcuna, and L. Cruz-Pizarro. 1994. The zooplankton biomass and food availability in four reservoirs of contrasting trophic status. Arch.Hydrobiol.Beih.,Ergebn.Limnol. 40:161-173.

Morin, A. 1985. Variability of density estimates and the optimization of sampling programs for stream benthos. Can.J.Fish.Aquat.Sci. 42:1530-1534.

Pace, M.L., 1984. Zooplankton community structure, but not biomass, influences the phosphorus-chlorophyll a relationship. Can.J.Fish.Aquat.Sci. 41:1089-1096.

Pepin, P. 1987. Comment on "Trophic relationships in freshwater pelagic ecosystems": a question of averages and sampling error? Can.J.Fish.Aquat.Sci. 44:1096-1097.

Peters, R.H. 1986. The role of prediction in limnology. Limnol.Oceanogr. 31:1143-1159.

Peterman, R.M. 1990. Statistical power analysis can improve fisheries research and management. Can.J.Fish.Aquat.Sci. 47:2-15.

Prairie, Y.T., C.M. Duarte, and J. Kalff. 1989. Unifying nutrient-chlorophyll relationships in lakes. Can.J.Fish.Aquat.Sci. 46:1176-1182.

Reckhow, K.H. 1994. Water quality simulation modeling and uncertainty analysis for risk assessment and decision making. Ecological Modelling 72:1-20.

Rohlf, F.J., and R.R. Sokal, 1981. Statistical Tables. 2nd ed. W.H. Freeman and Company, New York.

Sakamoto, M. 1966. Primary production by phytoplankton communities in some Japanese lakes and its dependence on lake depth. Arch. Hydrobiol. 62:1-28.

Smith, V.H. 1982. The nitrogen and phosphorus dependence of algal biomass in lakes: an empirical and theoretical analysis. Limnol. Oceanogr. 27:1101-1112.

Sokal, R.R., and F.J. Rohlf, 1981. Biometry, The Principles and Practice of Statistics in Biological Research. 2nd ed. W.H. Freeman and Company, New York.

Volleinweider, R.A. 1968. Water management research. OECD Paris. OECD/DAS/CSI/68.27.

Walmsley, R.D. 1984. A chlorophyll a trophic status classification system for South African impoundments. J. Environ. Qual. 13:97-104.

Zar, J.H. 1984. Biostatistical Analysis. 2nd ed. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

APPENDIX I

The maximum value for the S^2 of samples from a population

I use the following form of S^2 (Sokal and Rohlf, 1981):

$$S^2 = \frac{\Sigma(x_i^2) - n\bar{X}^2}{(n - 1)} \qquad [1]$$

where x_i is the magnitude of each individual measurement, $\bar{X}$ is the mean of all measurements, and n is the sample size.

For a given n and $\bar{X}$, S^2 is maximized by maximizing $\Sigma(x_i^2)$ because the other terms in [1] remain constant.

Maximizing $\Sigma(x_i^2)$

I use an approach that is relatively simple but not algebraically rigorous.

The samples are first arranged in n boxes. The focus will initially be on the last two boxes, boxes n-1 and n. Let their magnitudes be z and y so that subscripts are unnecessary. There are no negative magnitudes.

Now ask which arrangement yields a larger sum of squares:

the noncompiled boxes: $z^2 + y^2$ [2]

or the compiled boxes: $(z + y)^2 + 0^2$ [3]

[3] can be written as:

$$(z + y)^2 + 0^2 = \underline{z^2 + y^2} + \underline{2zy} \qquad [4]$$

Therefore the compilation of the two boxes gives the larger sum of squares, or an equal sum of squares when z or y are zero.

Next, one can ask what happens when those compiled boxes are compiled with box n-2. The same reasoning shows that an even higher $\Sigma(x_i^2)$ is obtained (except when one of the boxes has a magnitude of zero, then $\Sigma(x_i^2)$ remains unchanged). This process can be continued until all boxes are compiled into one. At that point, when no more compilations are possible, $\Sigma(x_i^2)$ is maximized. All but one box will have zero magnitude. The magnitude of the nonzero box will be Σx_i. If this value is inserted into [1]:

$$s^2 = \frac{(\Sigma x_i)^2 - n\bar{X}^2}{(n - 1)} \qquad [5]$$

$$= \frac{(n\bar{X})^2 - n\bar{X}^2}{(n - 1)}$$

$$= \frac{n^2\bar{X}^2 - n^2\bar{X}^2}{(n - 1)}$$

$$= \frac{n\bar{X}^2\,(n - 1)}{(n - 1)}$$

$$= n\bar{X}^2$$

Therefore the maximum s^2 is $n\bar{X}^2$.

APPENDIX II

Computer code for the CLT program

Note: The code is slightly more extensive than is required for this thesis. It is written in TBASIC.

```
REM Program that simulates scatter around a perfectly
REM deterministic linear model. Amount of scatter is related
REM to the estimated uncertainty involved in estimating the
REM population means - ie the data points - shown along
REM the x and y axis of a regression line.

cls
$stack 4054          'allocates extra memory for arrays

          '_____________Control Variables_____________

%wantGraphics     = 0  'non-zero value permits $IF/$ENDIF
                        block in SUB Regression
%wantDataPoints   = 0 'non-zero value permits $IF/$ENDIF
                       block in SUB Regression
%sampleSizeCheck = 0    'a loop to alter sample size, to see
                         effect on model SSE
%wantEachResult   = 0    'non-zero value causes samplesize
                          check to stop after each n.
%soundOn          = 0   'adds sound to alert user when
                         processing done.
wantSSEfile$ = "n"                'files modelSSE array.
wantSSEoverSSYfile$ = "n"         'files modelSSe/SSY array.
wantSSYfile$ = "n"                'files modelSSY array.
wantR2file$  = "n"                'files modelr2 array.
          '_____________________________________________

INPUT "maximum number of data points to be entered = ";
  maxNum%             'required for DIM statements
INPUT "number of simulated data sets requested = ";
  numTrials%

DIM DYNAMIC x(1:maxNum%), y(1:maxNum%), logX(1:maxNum%),
  logY(1:maxNum%), logRandx(1:maxNum%), logRandy(1:maxNum%),
  hypX(1:maxNum%), logHypX(1:maxNum%), hypY(1:maxnum%),
  logHypY(1:maxNum%)
DIM DYNAMIC unexplainedMSE(1:numTrials%),
  modelr2(1:numTrials%), pseudor2(1:numTrials%),
  arrayModelSSE(1:numTrials%),
  modelSSEoverSSY(1:numtrials%), arrayModelSSY(1:numtrials%)
DIM DYNAMIC Xmax(1:maxNum%), Xmin(1:maxNum%),
  Ymax(1:maxNum%), Ymin(1:maxNum%),
```

```
  repLevelSlope(1:maxNum%), repLevelIntercept(1:maxNum%),
  repLevelS2YgivenX(1:maxNum%), S2Y(1:maxNum%),
  S2X(1:maxNum%)
DIM DYNAMIC sampleSizeX%(1:maxNum%),
  sampleSizeY%(1:maxNum%), lowerCI(1:maxNum%),
  upperCI(1:maxNum%), logPerfectFitY(1:maxNum%)
  '******temporarily here to test graphics
DIM DYNAMIC S2hypX(1:maxNum%), S2hypY(1:maxNum%)
DIM DYNAMIC sortedlogRealX(1:maxNum%),
  sortedlogRealY(1:maxNum%), sortedlogRandX(1:maxNum%),
  sortedXmin(1:maxNum%), sortedXmax(1:maxNum%),
  sortedYmin(1:maxNum%), sortedYmax(1:maxNum%)
DIM DYNAMIC orthoX(1:maxNum%), orthoY(1:maxNum%),
  Xgap(1:maxNum%), Ygap(1:maxNum%)

simulationsForX& = 0   'increments at each simulation.
simulationsForY& = 0

meanXboundByHi95CI&    = 0  'initializes variable to
                             count occurances of a simulated
                             meanX bound by 95% CI.
meanXboundByLo95CI&    = 0   'bound by 95% CI.
meanXboundByMin&       = 0   'bound by preset minimum value.
meanXboundByMax&       = 0   'bound by preset maximum value.

minMeanXisUnder95CI& = 0  'occurances of: preset min
                           values that are less than 95% CI.
maxMeanXisOver95CI&  = 0    'max values that are greater
                             than 95% CI.

meanYboundByHi95CI&    = 0  'as above, for simulated meanY.
meanYboundByLo95CI&    = 0
meanYboundByMin&       = 0
meanYboundByMax&       = 0

minMeanYisUnder95CI& = 0
maxMeanYisOver95CI&  = 0

repxBoundByHi95CI&     = 0    'initializes variable to count
                               occurances of a simulated
                               replicate x bound by 95% CI.
repxBoundByLo95CI&     = 0    'bound by 95% CI.
repxBoundByMin&        = 0    'bound by preset minimum value.
repxBoundByMax&        = 0    'bound by preset maximum value.

minRepxIsUnder95CI& = 0    'occurances of: preset min values
                            that are less than 95% CI.
maxRepxIsOver95CI&  = 0    'max values that are greater than
                            95% CI.

repyBoundByHi95CI&     = 0    ' as above, for simulated
replicate y.
```

```
repyBoundByLo95CI&      = 0
repyBoundByMin&         = 0
repyBoundBYMax%         = 0

minRepyIsUnder95CI& = 0
maxRepyIsOver95CI&  = 0


CALL GETDATA ( sampleSizeX%(), sampleSizeY%(), numTrials%,
  numPoints%, X(), Y(), logX(), logY())
CALL GETS2INFO ( S2method%, S2X(), S2Y(), numPoints%,
  replicateCorrelation$, expYS2, propConstYS2, expXS2,
  propConstXS2, repLeveLSlope(), repLevelIntercept(),
  repLevelS2YgivenX(), maxS2X, minS2X, maxS2Y, minS2Y)
CALL GetBoundsInfo (numPoints%, boundsType%, repBoundsType%,
  Xmax(), Xmin(), Ymax(),Ymin(), minRepx, maxRepx, minRepy,
  maxRepy)
CALL Regression (logX(),logY(), (numPoints%), realSSY,
  realSSE, realSlope, realIntercept)
CALL HypothesizedXY (logHypX(), hypX(), logHypY(), hypY(),
  S2hypX(), S2hypY())
CALL MinMaxCIcheck ("meanX", X(), Xmin(), Xmax(), hypX(),
  sampleSizeX%(), S2X(), S2hypX())
CALL MinMaxCIcheck ("meanY", Y(), Ymin(), Ymax(), hypY(),
  sampleSizeY%(), S2Y(), S2hypY())

startTime1& = TIMER   'initial time, used to time
                       non-interactive part of program.
loopCounter& = 1   'used in counting %sampleSizeCheck loops

         $IF %sampleSizeCheck      'entered only if turned on
                                    by user
             PRINT "The sample size program uses a common
               sample size for X and Y."
             PRINT " "
             INPUT " average sample size used = ";
               meanSampleSize%
             DO  'end of DO is embedded in $IF/$ENDIF below
               cls
               IF loopCounter& > 1 THEN
                  PRINT "Sample size just tested:    ";
                    meanSampleSize%
                  PRINT "
                  PRINT "probability modelSSE > realSSE = ";
                    probModelSSEgtRealSSE
               END IF

               meanSampleSize% = meanSampleSize% + 1
               FOR i = 1 to numPoints%
                 sampleSizeX%(i) = meanSampleSize%
                 sampleSizeY%(i) = meanSampleSize%
               NEXT i
```

```
        $ENDIF

FOR trial = 1 to numTrials%
   IF replicateCorrelation$ = "y" THEN
      CALL SimulateWhenReplicatesCorr ( sampleSizeX%(),
        sampleSizeY%(), numPoints%, x(), expXS2,
        propConstXS2, hypY(), expYS2, propConstYS2,
        repLevelSlope(), repLevelIntercept(),
        repLevelS2YgivenX(), logRandX(), logRandY())
   ELSE
      CALL SimulateSamples ( hypY(), sampleSizeX%(),
        sampleSizeY%(), (numPoints%), (expXS2),
        (propConstXS2), (expYS2), (propConstYS2), hypX(),
        logRandX(), logRandY())
   END IF
   CALL Regression (logRandx(), logRandy(), (numPoints%),
     modelSSY, modelSSE, slope, intercept)

        $IF %wantGraphics
           CALL Graphics (boundsType%, Xmin(), Xmax(),
             Ymin(), Ymax(), numPoints%, logX(), logY(),
             realSlope, realIntercept, logRandX(),
             logRandY(), slope , intercept)
        $ENDIF
   CALL CompileSSEdist ((modelSSE), (modelSSY), (realSSE),
     (realSSY), (numPoints%), (trial), unexplainedMSE(),
     modelr2(), pseudor2(), arrayModelSSE(),
     modelSSEoverSSY(), arrayModelSSY() )
NEXT trial

IF %sampleSizeCheck=0 THEN
   finishTime1& = TIMER

   $IF %soundOn
      CALL SoundAlert(nullVariable)
   $ENDIF

   CALL PrintBoundsInfringements ( numPoints%, numTrials%,
     boundsType%, meanXboundByMin&, meanXboundByMax&,
     meanXboundByHi95CI&, meanXboundByLo95CI&,
     minMeanXisUnder95CI&, maxMeanXisOver95CI&,_

     meanYboundByMin&, meanYboundByMax&,
     meanYboundByHi95CI&, meanYboundByLo95CI&,
     minMeanYisUnder95CI&, maxMeanYisOver95CI&)
   IF replicateCorrelation$ = "y" THEN
     CALL ReplicateBoundsInfringements ( repBoundsType%,
       repxBoundByMin&, repxBoundByMax&, repxBoundByHi95CI&,
       repxBoundByLo95CI&, minRepxIsUnder95CI&,
       maxRepxIsOver95CI&,_

       repyBoundByMin&, repyBoundbyMax&, repyBoundByHi95CI&,
```

```
        repyBoundByLo95CI&, minRepyIsUnder95CI&,
        maxRepyIsOver95CI&)
    END IF
END IF

startTime2& = TIMER  'restarts program timing after user
                      interaction in SUB
                      PrintBoundsInfringements

CALL ComputeSSEstats ( unexplainedMSE(), modelr2(),
  pseudor2(), (numTrials%), meanUnexplainedMSE, meanModelr2,
  meanPseudor2, arrayModelSSE(), (realSSE),
  probModelSSEgtRealSSE, meanModelSSE, lowerBoundMeanSSE,
  upperBoundMeanSSE)
CALL FindProbModelSSEgtRealSSE ( (realSSE), (numTrials%),
  arrayModelSSE(), probModelSSEgtRealSSE)

IF CIwarning$ = "y" THEN PRINT "WARNING CI below determined
  from too few random trials."
PRINT " "
PRINT "                                RESULTS:"
PRINT "                      (with 95% CI for simulations)"
PRINT "
-----------------------------------------------------------
----------"
PRINT USING "| realSSE                                    |
   ###.####             |"; realSSE
PRINT USING "| mean modelSSE                              |    ###.####
  ! ###.#### ! ###.#### |"; lowerBoundMeanSSE,
  "-",meanModelSSE, "-", upperBoundMeanSSE
PRINT       "|                                            |
                        |"
PRINT USING "| prob. that modelSSE >= realSSE             |
     #.###               |"; probModelSSEgtRealSSE
PRINT "
-----------------------------------------------------------
----------"
PRINT " "
PRINT "
-----------------------------------------------------------
----------"
PRINT USING "| mean sq. error of real data                |
   ###.####             |"; realSSE/numPoints%   '****check
   formula
PRINT USING "| mean unexplained mean sq. error            |    ###.####
  ! ###.#### ! ###.#### |"; lowerBoundMSE, "-",
  meanUnexplainedMSE, "-", upperBoundMSE
PRINT "
-----------------------------------------------------------
----------"
PRINT " "
PRINT "
```

```
_________"
PRINT USING "| real r2                                    |
     #.##                |"; (realSSY - realSSE) / realSSY
PRINT USING "| mean modelr2                               |
  #.## ! #.## ! #.##        |"; lowerBoundModelr2, "-",
  meanModelr2, "-", upperBoundModelr2
PRINT USING "| mean pseudor2                              |
  #.## ! #.## ! #.##        |"; lowerBoundPr2, "-",
  meanPseudor2, "-", upperBoundPr2
PRINT "
-------------------------------------------------------------
---------"
PRINT "   "

finishTime2& = TIMER

            $if %sampleSizeCheck

              $IF %wantEachResult
                 input "press any key to continue";carryon$
              $ENDIF

              INCR loopCounter&
              LOOP UNTIL probModelSSEgtRealSSE <= .05  OR
                meanSampleSize% >= 100

              $IF %soundOn
                 CALL SoundAlert (nullVariable)
              $ENDIF

              INPUT "Samplesize looping completed. Press any
                key for bounds information."; carryOn$
              cls
              loopCounter& = loopCounter& - 1   'resets to
                                                 appropriate
                                                 value
              CALL PrintBoundsInfringements ( numPoints%,
                numTrials%, boundsType%, meanXboundByMin&,
                meanXboundByMax&, meanXboundByHi95CI&,
                meanXboundByLo95CI&, minMeanXisUnder95CI&,
                maxMeanXisOver95CI&,_

                meanYboundByMin&, meanYboundByMax&,
                meanYboundByHi95CI&, meanYboundByLo95CI&,
                minMeanYisUnder95CI&, maxMeanYisOver95CI&)
              IF replicateCorrelation$ = "y" THEN
                 CALL ReplicateBoundsInfringements (
                   repBoundsType%, repxBoundByMin&,
                   repxBoundByMax&, repxBoundByHi95CI&,
                   repxBoundByLo95CI&, minRepxIsUnder95CI&,
                   maxRepxIsOver95CI&,_
```

```
                repyBoundByMin&, repyBoundbyMax&,
                repyBoundByHi95CI&, repyBoundByLo95CI&,
                minRepyIsUnder95CI&, maxRepyIsOver95CI&)
            END IF

            IF meanSampleSize% >= 100 THEN
               PRINT "required sample size exceeds 100;
                 program stopped."
            ELSE
               PRINT " "
               PRINT; "required sample size is = ";
                 meanSampleSize%
            END IF
            PRINT " "
            elapsedTime = (finishTime2& - startTime1&) / 60
            PRINT USING " THAT'S ALL: elapsed time = ####.#
              &"; elapsedTime, " min."
            END
          $ENDIF
elapsedTime1 =  (finishTime1& - startTime1&) / 60
elapsedTime2 =  (finishTime2& - startTime2&) / 60
PRINT USING "THAT'S ALL: elapsed time  PART 1 = ####.# &
  ####.# &"; elapsedTime1, " min., PART 2 = ", elapsedTime2,
  "min."
END

rename:     'file error trapping
INPUT "file not found, please try again"; fileName$
RESUME

'-------------------------------------------------------------
----------
'                              PROCEDURES
'-------------------------------------------------------------
----------


SUB GetData ( sampleSizeX%(1), sampleSizeY%(1), numTrials%,
  numPoints%, x(1), y(1), logx(1), logy(1))
REM retrieves measured data points from file or keyboard.
SHARED fileName$     'this allows file error trapping (see
                      RESUME above)
LOCAL i, continue$, xRawDataLogged$, yRawDataLogged$,
  sampleSize%, xSampleSize%, ySampleSize%

INPUT "Is x raw data log transformed? Use small letters
  y/n. ";xRawDataLogged$
INPUT "Is y raw data log transformed? Use small letters
  y/n. ";yRawDataLogged$
PRINT " "
INPUT "Is XY Data on file or to be keyboarded? Use small
```

```
  letters  f/k. "; route$
IF route$ = "f" THEN
   INPUT "What is the XY data file name"; fileName$
   CALL GetFile ( fileName$, x(), y(), numPoints% )
ELSE
   CALL GetKeyboard ( x(), y(), numPoints%)
END IF

IF xRawDataLogged$ = "y" THEN
   FOR i = 1 to numPoints%
     logx(i)=x(i)                    'x values already logged
     x(i)=10^logx(i)
   NEXT
ELSE                                 'transform data
   FOR i = 1 to numPoints%
     logx(i)=LOG10(x(i))
   NEXT
END IF
IF yRawDataLogged$ = "y" THEN
   FOR i = 1 to numPoints%           'y values already logged
     logy(i)=y(i)
     y(i)=10^logy(i)
   NEXT
ELSE                                 'transform data
   FOR i = 1 to numPoints%
     logy(i)=LOG10( y(i) )
   NEXT
END IF

PRINT " "
PRINT "    X              Y"
PRINT "---------     ---------"

FOR i = 1 to numPoints%
   PRINT USING "####.###       ####.###"; x(i), y(i)
   IF ((i/15) - FIX(i/15)) = 0 THEN
       INPUT "more, press any key";carryOn$
   END IF
NEXT
PRINT " "
INPUT "press any key to continue";carryOn$
cls


PRINT "Options for sample sizes:"
PRINT "-------------------------"
PRINT " "
PRINT " 1. The same sample size is used for all X and Y."
PRINT " 2. Sample size varies with location, but is the same
  for each (X,Y)."
PRINT " 3. One sample size for all X, a different sample
  size for all Y."
```

```
PRINT " 4. Each sample size must be entered separately."
PRINT " "
INPUT " Which would you like: 1,2,3, or 4"; sampleSizeType%
cls

SELECT CASE sampleSizeType%
  CASE 1
     PRINT "The same sample size is used for all X and Y."
     PRINT " "
     INPUT "sample size = "; sampleSize%

     FOR i = 1 to numPoints%
       sampleSizeX%(i) = sampleSize%
       sampleSizeY%(i) = sampleSize%
     NEXT i

  CASE 2
     PRINT "Sample size varies with location, but is the
same for each (X,Y)."
     INPUT "Are sample size data on file (y/n)";file$
     IF file$ = "y" THEN
        INPUT "what is the file name ";fileName$
        CALL Geta2variableFile (fileName$, sampleSizeX%(),
          sampleSizeY%())
     ELSE
        PRINT " "
        PRINT "Each location's sample size must be entered
          separately."
        PRINT " "
        PRINT "Location: "
        PRINT "---------"

        FOR i = 1 to numPoints%
          PRINT; i;"."
          INPUT "sample size = "; sampleSizeX%(i)
          sampleSizeY%(i) = sampleSizeX%(i)
        NEXT i
     END IF

  CASE 3
     PRINT "One sample size for all X, a different sample
       size for all Y."
     PRINT " "
     INPUT "X sample size = "; xSampleSize%
     INPUT "Y sample size = "; ySampleSize%
     PRINT " "
     PRINT "Location:"
     PRINT "---------"

     FOR i = 1 to numPoints%
        sampleSizeX%(i) = xSampleSize%
        sampleSizeY%(i) = ySampleSize%
```

```
      NEXT i

   CASE 4
      INPUT "Are sample size data on file (y/n)";file$
      IF file$ = "y" THEN
         INPUT "what is the file name ";fileName$
         CALL Geta2variableFile (fileName$, sampleSizeX%(),
           sampleSizeY%())
      ELSE
         PRINT "Each sample size must be entered separately."
         PRINT " "
         PRINT "Location:"
         PRINT "---------"
         FOR i = 1 to numPoints%
            PRINT; i;"."
            INPUT "X sample size = "; sampleSizeX%(i)
            INPUT "Y sample size = "; sampleSizeY%(i)
         NEXT i
      END IF

   CASE OTHER
       PRINT "ERROR. SAMPLE SIZE TYPE NOT SELECTED PROPERLY."
       PRINT "THERE IS AN ERROR CHECK AHEAD; PLEASE, RE-ENTER
         ANY DATA REQUESTED."
       INPUT "press any key"; carryOn$
END SELECT
cls

PRINT "Sample Sizes:"
PRINT " X         Y
PRINT "----      ----"
PRINT " "

FOR i = 1 to numPoints%
   PRINT USING "####      ####"; sampleSizeX%(i),
sampleSizeY%(i)
NEXT i
PRINT " "
INPUT "Would you like this data filed (y/n)";fileMe$
IF fileMe$ = "y" THEN
   INPUT "What name would you like (DOS restrictions)";
fileName$
   CALL Create2variableFile (fileName$, numPoints%,
sampleSizeX%(), sampleSizeY%())
END IF

INPUT "Error check. Continue (y/n)";continue$
cls

IF continue$ = "n" THEN
   CALL  GETDATA ( sampleSizeX%(), sampleSizeY%(),
     numTrials%, numPoints%, x(), y(), logx(), logy())
```

```
END IF
END SUB


SUB Geta2variableFile (moniker$, a%(1), b%(1))
REM siphons a 2 integer variable DOS file.
LOCAL count%, carryOn$

ON ERROR GOTO rename
OPEN moniker$ FOR INPUT AS #1       'accesses an external
                                     data file
ON ERROR GOTO 0
$EVENT OFF

count% = 0
DO UNTIL EOF(1)
  count% = count% + 1     'count% will have a count upon exit
  INPUT #1, a%(count%), b%(count%)
LOOP
CLOSE #1
PRINT " "
PRINT "Data read from "; moniker$
PRINT "file has ";count%;" observations."
INPUT "press any key to continue";carryOn$
cls
END SUB


SUB Create2VariableFile (moniker$, numPoints%, a%(1), b%(1))
REM creates a 2 integer variable external data file.
LOCAL  count%, carryOn$

OPEN moniker$ FOR OUTPUT AS #1
count% = 0
DO
  count% = count% + 1
  PRINT #1, a%(count%), b%(count%)
LOOP UNTIL count% = numPoints%
CLOSE #1
PRINT "File "; moniker$;" created."
INPUT "press any key to continue";carryOn$
cls
END SUB


SUB GetS2Info ( S2method%, S2X(1), S2Y(1), numberOfLakes%,
  replicateCorrelation$, expYS2, propConstYS2, expXS2,
  propConstXS2, repLeveLSlope(1), repLevelIntercept(1),
  repLevelS2YgivenX(1), maxS2X, minS2X, maxS2Y, minS2Y)
```

```
REM gets S2 information from operator.
SHARED X(), Y(), fileName$    'allows file name error
                               trapping
LOCAL setS2Limits$, carryOn$

PRINT "Options for estimating S2 associated with meanX and
  meanY:"
PRINT
"-------------------------------------------------------"
PRINT " "
PRINT " 1. Use a mean:variance function."
PRINT " "
PRINT " 2. Use S2 measured from each lake."
PRINT " "
INPUT " Which would you like: 1,2"; S2method%
cls

SELECT CASE S2method%
  CASE 1                        'mean:variance regression used
     PRINT " "
     INPUT "Exponent, b, of X S2 predictor function
       (S2=aX^b) = ";expXS2
     INPUT "value of  a  from S2= a X^b    = ";propConstXS2
     PRINT " "
     PRINT " "
     INPUT "Exponent, b, of Y S2 predictor function
       (S2=aY^b) = ";expYS2
     INPUT "value of a from S2= a Y^b    = ";propConstYS2

      FOR i = 1 to numberOfLakes%
         S2X(i) = propConstXS2 * (X(i) ^ expXS2)
         S2Y(i) = propConstYS2 * (Y(i) ^ expYS2)
      NEXT
      cls

  CASE 2                  'use S2 measured in each lake.
     PRINT "You have chosen to use different S2 values for
       each lake."
     INPUT "Are S2 data on file (y/n)"; onFile$
     IF onFile$ = "y" THEN
        INPUT "What is filename"; fileName$
        CALL GetFile (fileName$, S2X(), S2Y(),
          numberOfLakes%)
     ELSE
        FOR  i = 1 to numberOfLakes%
          PRINT " "
          PRINT " "
          PRINT "Lake ";i
          PRINT "----------"
          PRINT " "
          INPUT " S2 x = "; S2X(i)
          PRINT " "
```

```
          INPUT " S2 y = "; S2Y(i)
        NEXT i
        cls
      END IF
END SELECT

CALL Print2RealVariables ( "  X S2          ", "   Y S2 ",
  S2X(), S2Y(), numberOfLakes%)

INPUT "Is there correlation betwen x and y at the replicate
  level (y/n)"; replicateCorrelation$
IF replicateCorrelation$ = "y" THEN
   PRINT " "
   PRINT "Is there a file holding:"
   INPUT "replicate level slope, intercept and and S2y|x
     (y/n)"; repFile$
   IF repFile$ = "y" THEN
      INPUT "What is the file name "; fileName$
      cls
      CALL GetRepParameters (fileName$, repLevelSlope(),
        repLevelIntercept(), repLevelS2YgivenX())
   ELSE                          ' ie. repFile$ = "n"
      PRINT " "
      PRINT " Parameter values required from replicate
        regression (rep.x vs. rep.y):"
      PRINT
"-------------------------------------------------------------
------------"
      PRINT " "
      FOR i=1 to numberOfLakes%
          PRINT " "
          PRINT " "
          PRINT "     Lake ";i
          PRINT "     ---------------------------------"
          PRINT " "
          INPUT "     slope       = "; repLevelSlope(i)
          INPUT "     intercept   = "; repLevelIntercept(i)
          INPUT "     S2Y given X = "; repLevelS2YgivenX(i)
      NEXT i
   END IF                        'repFile$
   cls
   PRINT " "
   PRINT " "
   PRINT "Lake         slope          intercept          S2y
     given x "
   PRINT
"-------------------------------------------------------------
   PRINT " "

   FOR i = 1 to numberOfLakes%
      PRINT USING "####       ####.####       ####.####
        ####.####";i, repLevelSlope(i),
```

```
        repLevelIntercept(i), repLevelS2YgivenX(i)
      IF INT(i/15) - (i/15) = 0 THEN INPUT "More, press any
        key";carryOn$
   NEXT i

   INPUT "Would you like this data to be filed  (y/n) ";
     fileRepParameters$

   IF fileRepParameters$ = "y" THEN
     INPUT "What file name would you like (DOS restrictions)
       "; fileName$
     CALL CreateRepParameterFile (filename$, numberOfLakes%,
       repLevelSlope(), repLevelIntercept(),
       repLevelS2YgivenX())
   END IF
END IF              'replicateCorrelation$

PRINT " "
INPUT "Error check. Continue (y/n)";continue$
cls
IF continue$ = "n" THEN
   CALL GETS2INFO ( S2method%, S2X(), S2Y(), numberOfLakes%,
     replicateCorrelation$, expYS2, propConstYS2, expXS2,
     propConstXS2, repLeveLSlope(), repLevelIntercept(),
     repLevelS2YgivenX(), maxS2X, minS2X, maxS2Y, minS2Y)
END IF
END SUB



SUB GetRepParameters (moniker$, slope(1), intercept(1),
  S2YgivenX(1))
REM siphons rep parameters from a DOS file.
LOCAL count%

$EVENT ON
ON ERROR GOTO rename
OPEN moniker$ FOR INPUT AS #1       'accesses an external
                                     data file
ON ERROR GOTO 0
$EVENT OFF

  count% = 0
  DO UNTIL EOF(1)
    count% = count% + 1  'count% will have a count upon exit
    INPUT #1, slope(count%), intercept(count%),
      S2YgivenX(count%)
  LOOP
CLOSE #1
PRINT " "
PRINT "Data read from "; moniker$
PRINT "file has ";count%;" observations."
```

```
INPUT "press any key to continue";carryOn$
cls
END SUB


SUB CreateRepParameterFile (moniker$, numPoints%, slope(1),
  intercept(1), S2YgivenX(1))
REM creates an external data file for replicate correlation
  info.
LOCAL  count%

OPEN moniker$ FOR OUTPUT AS #1
count% = 0
DO
  count% = count% + 1
  PRINT #1, slope(count%), intercept(count%),
    S2YgivenX(count%)
LOOP UNTIL count% = numPoints%
CLOSE #1
PRINT "File "; moniker$;" created."
INPUT "press any key to continue";carryOn$
cls
END SUB


SUB GetBoundsInfo (numberOfLakes%, boundsType%,
  repBoundsType%, Xmax(1), Xmin(1), Ymax(1), Ymin(1),
  minRepx, maxRepx, minRepy, maxRepy)
REM more data requirements, these are in two SUBs because of
  tbasic restrictions.
LOCAL  continue$, minMeanX, maxMeanX, minMeanY, maxMeanY
SHARED replicateCorrelation$, fileName$

PRINT "Options for constraining simulated mean values:"
PRINT "-----------------------------------------------"
PRINT " "
PRINT " Values will be bound by:"
PRINT " "
PRINT " 1.  95%CI and a min value."
PRINT " 2.  95%CI and one min and max value shared by all
  lakes."
PRINT " 3.  a min and max value unique to each lake."
PRINT " "
INPUT " Which would you like: 1,2,3 "; boundsType%
cls

Select Case boundsType%
  Case 1
     PRINT " "
     PRINT "Simulated values of meanX and meanY will be
```

```
      bound by 95% CI,"
    PRINT "also simulated values will be bound by an
      arbitrary
    PRINT "min value."
    PRINT " "
    INPUT " min meanX value: "; minMeanX
    PRINT " "
    INPUT " min meanY value: "; minMeanY
    FOR i = 1 to numberOfLakes%
       Xmin(i) = minMeanX
       Ymin(i) = minMeanY
    NEXT
    cls
    PRINT "**** max not applicable ****"
Case 2
    PRINT " "
    PRINT "Simulated values of meanX and meanY will be
      bound by 95% CI,"
    PRINT "and by a general max or min value."
    PRINT " "
    INPUT " min meanX value: "; minMeanX
    INPUT " max meanX value: "; maxMeanX
    IF (minMeanX >= maxMeanX) THEN CALL BoundsErr
      (minMeanX, maxMeanX)
    PRINT " "
    INPUT " min meanY value: "; minMeanY
    INPUT " max meanY value: "; maxMeanY
    IF (minMeanY >= maxMeanY) THEN CALL BoundsErr
      (minMeanY, maxMeanY)
    FOR i = 1 to numberOfLakes%
       Xmin(i) = minMeanX
       Xmax(i) = maxMeanX
       Ymin(i) = minMeanY
       Ymax(i) = maxMeanY
    NEXT
    cls
CASE 3
    PRINT " "
    PRINT "Simulated values of meanX and meanY will be
      bound by 95% CI, "
    PRINT "and by a unique max or min value."
    PRINT " "
    INPUT "Are min-max data on file (y/n)";file$
    IF file$ = "y" THEN
      INPUT "What is file name for X min-max
        data";fileName$
      CALL GetFile (fileName$, Xmin(), Xmax(),
        numberOfLakes%)
      INPUT "What is the file name for Y min-max data";
        fileName$
      CALL GetFile (fileName$, Ymin(), Ymax(),
        numberOfLakes%)
```

```
      ELSE
        PRINT " "
        PRINT "First, the x values:"
        PRINT " "

        FOR i=1 to numberOfLakes%
          PRINT "for lake ";i
          INPUT "min replicate x value "; Xmin(i)
          INPUT "max replicate x value "; Xmax(i)
          IF (Xmin(i) >= Xmax(i))  THEN CALL BoundsErr
            (Xmin(i), Xmax(i))
          PRINT " "
        NEXT

        PRINT "now, the y values: "
        PRINT " "

        FOR i=1 to numberOfLakes%
          PRINT "for lake ";i
          INPUT "min replicate y value "; Ymin(i)
          INPUT "max replicate y value "; Ymax(i)
          IF (Ymin(i) >= Ymax(i))  THEN CALL BoundsErr
            (Ymin(i), Ymax(i))
          PRINT " "
        NEXT
      END IF
      cls
END SELECT
      PRINT" "
      PRINT "Lake        X (min - max)                  Y(min -
        max)"
      PRINT
"-------------------------------------------------------"
       a$ = "###     !####.#### ! ####.####!     !####.#### !
         ####.####!"

      FOR i=1 to numberOfLakes%
         IF ((i/15) - FIX(i/15)) = 0 THEN
            INPUT "more, press any key";carryOn$
         END IF
         PRINT USING a$;i,"(", Xmin(i),"-", Xmax(i),")","(",
           Ymin(i),"-", Ymax(i),")"
      NEXT i

      PRINT " "
      INPUT "Would you like these data filed (y/n)"; fileIt$
      IF fileIt$ = "y" THEN
         INPUT "Please name Xmin,max file"; fileName$
         CALL CreateFile (fileName$, Xmin(), Xmax(),
           numberOfLakes%)
         INPUT "Please name Ymin,max file"; fileName$
         CALL CreateFile (filename$, Ymin(), Ymax(),
```

```
          numberOfLakes%)
      END IF

INPUT "Error check. Continue (y/n)"; continue$
cls
IF continue$ = "n" THEN
   CALL GetBoundsInfo (numberOfLakes%, boundsType%,
     repBoundsType%, Xmax(), Xmin(), Ymax(), Ymin(),
     minRepx, maxRepx, minRepy, maxRepy)
END IF
cls

IF replicateCorrelation$ = "y" THEN
   $INCLUDE "repdat.inc"        'code is in another file to
     free up editor memory. *** not in this appendix ***
END IF                 'replicateCorrelation$ ="y"
END SUB



SUB BoundsErr (min, max)
REM traps input errors for two real variables, allows
  re-entry.
PRINT " "
PRINT "input error: a min value equaled or exceeded a max.
  Please try again."
PRINT " "
INPUT " min value: "; min
INPUT " max value: "; max
IF (min >= max) THEN CALL BoundsErr (min, max)
END SUB



SUB GetFile (moniker$, a(1), b(1), count%)
REM  gets a 2 variable file from disk, counts records.
LOCAL i, carryOn$

$EVENT ON
ON ERROR GOTO rename
OPEN moniker$ FOR INPUT AS #1       'accesses an external
                                     data file
ON ERROR GOTO 0
$EVENT OFF

  count% = 0
  DO UNTIL EOF(1)
    count% = count% + 1
    INPUT #1, a(count%), b(count%)
  LOOP
```

```
CLOSE #1
PRINT " "
PRINT "Data read from ";moniker$
PRINT "file has ";count%;" data pairs."
INPUT "Press any key to continue";carryOn$
cls
END SUB


SUB GetKeyboard ( x(1), y(1), numPoints%)
REM  takes values from keyboard, places on disk if desired.
SHARED xRawDataLogged$, yRawDataLogged$
LOCAL i, makeFile$

INPUT "Number of data points to be entered = "; numPoints%
FOR i=1 to numPoints%
   PRINT " "
   PRINT  "observation "; i
   PRINT  "-----------------"
   INPUT "X value  = ";x(i)
   INPUT "Y value  = ";y(i)
NEXT
IF (xRawDataLogged$ = "y" OR yRawDataLogged$ = "y") THEN
'when data are not logged then filing takes place in GetData
   PRINT " "
   INPUT "transfer data to a file? Use small letters (y/n)."
     ; makeFile$
   IF makeFile$ = "y" THEN
      INPUT "What is file name (DOS restrictions)";fileName$
      CALL CreateFile ( fileName$, x(), y(), (numPoints%))
   END IF
END IF
cls
END SUB


SUB CreateFile (moniker$, a(1), b(1), numPoints%)
REM creates an external data file
LOCAL n%, carryOn$

OPEN moniker$ FOR OUTPUT AS #1
  n% = 0
  DO
    n% = n% + 1
    PRINT #1, a(n%), b(n%)
  LOOP UNTIL n% = numPoints%
CLOSE #1
PRINT "File "; moniker$;" created."
INPUT "Press any key to continue";carryOn$
cls
```

```
END SUB


SUB Regression (x(1), y(1), n%, SSY, SSE, slope, intercept)
REM  takes in log transformed values and regresses y on x.
LOCAL i, sumx, sumy, meanx, meany, SSX, preCovXY,
  predictedY, carryOn$, pleaseFile$
SHARED trial

SSY=0
SSE=0

FOR i = 1 to n%
   sumx = sumx + x(i)
   sumy = sumy + y(i)
NEXT

meanX = sumx / n%
meanY = sumY / n%

FOR i = 1 to n%
   SSX = SSX + ( x(i) - meanX ) ^ 2
   SSY = SSY + ( y(i) - meanY ) ^ 2
   preCovXY = preCovXY + ( x(i) - meanX ) * ( y(i) - meanY )
NEXT

slope = preCovXY / SSX
intercept = meanY - (slope * meanX)

FOR i = 1 to n%
   predictedY = slope * x(i) + intercept
   SSE = SSE + ( predictedY  -   y(i))^2
NEXT

$IF %wantDataPoints
    PRINT "SIMULATION #";trial
    PRINT "-------------------------"
    PRINT USING "slope     = ###.###"; slope
    PRINT USING "intercept = ###.###"; intercept
    PRINT USING "SSE       = ###.###"; SSE
    PRINT USING "r2        = ###.###"; (SSY - SSE)/SSY
    PRINT " "
    CALL Print2RealVariables ( " logRandX       ", "
      logRandY", X(), Y(), n%)
$ENDIF
END SUB


SUB HypothesizedXY (logHypX(1), hypX(1), logHypY(1),
  hypY(1), S2hypX(1), S2hypY(1))
```

```
REM: Puts hypothesized XY values into hypX, hypY. (These
  values are later used
REM: to simulate sampling, and then results are compared
  with real data SSE).
SHARED fileName$, numPoints%, X(), logX(), realSlope,
  realIntercept, Y(), logY(), expXS2, propConstXS2, expYS2,
  propConstYS2, S2Method%, S2X(), S2Y(), orthoX(), orthoY(),
  Xgap(), Ygap()    'orthoX & Y() and gapX & Y() here to
                     avoid DIM
LOCAL fitType%, i, retry$, factor

PRINT "Choices for Fitting meanX & meanY:"
PRINT "---------------------------------"
PRINT " "
PRINT " "
PRINT "1. use measured meanX, and meanY fitted to log:log
  regression line."
PRINT "2. ... fitted orthogonally to regression line."
PRINT "3. ... fitted part way along orthogonal from
  regression line to real data point."
PRINT "4. use a high SSE (meanX,meanY) data set."
PRINT "
PRINT " "
INPUT "Please choose a number: 1,2,3, or 4";fitType%
cls

SELECT CASE fitType%
  CASE 1
     FOR i = 1 to numPoints%
        logHypY(i) = realSlope * logX(i) + realIntercept
        hypY(i) = 10 ^ logHypY(i)
        hypX(i) = X(i)
        logHypX(i) = logX(i)
     NEXT

  CASE 2
     orthoSlope = -(1 / realSlope)
     FOR i = 1 to numPoints%
        orthoIntercept =logY(i) - (logX(i) * orthoSlope)
          'ie.  b = y - Mx
        loghypX(i) = (orthoIntercept - realIntercept) *
          (realSlope / ((realSlope ^ 2) + 1))
          'derived from 1) y=-(1/m)x + bortho  2) y=mx +
           breal
        logHypY(i) = orthoSlope * logHypX(i) +
          orthoIntercept
        hypX(i) = 10 ^ loghypX(i)
        hypY(i) = 10 ^ logHypY(i)
     NEXT

  CASE 3
     INPUT "enter the factor (0-1: proportional distance
```

```
        towards real data along orthogoanl line)"; factor
      orthoSlope = -(1 / realSlope)
      FOR i = 1 to numPoints%
         orthoIntercept =logY(i) - (logX(i) * orthoSlope)
           'ie.  b = y - Mx
         orthoX(i) = (orthoIntercept - realIntercept) *
           (realSlope / ((realSlope ^ 2) + 1))
           'derived from 1) y=-(1/m)x + bortho  2) y=mx +
            breal
         orthoY(i) = orthoSlope * orthoX(i) + orthoIntercept
           'orthogoanl projections to regression line.
         Xgap(i) = factor * (logX(i) - orthoX(i))
           'distance desired along orthogonl
         Ygap(i) = factor * (logY(i) - orthoY(i))
         logHypX(i) = orthoX(i) + Xgap(i)
         logHypY(i) = orthoY(i) + Ygap(i)
         hypX(i) = 10 ^ loghypX(i)
         hypY(i) = 10 ^ logHypY(i)
      NEXT

   CASE 4
      PRINT "A sort-of nul test will be performed: sampling
        will be simulated"
      PRINT "using high SSE X,Y (rather than X,Y along
        regression line). This"
      PRINT "will show how often real data X,Y might be
        expected from high SSE X,Y."
      PRINT " "
      INPUT "What is the file name for high SSE (logged) data
        ";fileName$
      CALL  GetFile (fileName$, logHypX(), logHypY(),
        numPoints%)
      FOR i = 1 to numPoints%
        hypY(i) = 10 ^ (logHypY(i))
        hypX(i) = 10 ^ (logHypX(i))
      NEXT

   CASE OTHER
      reTry$ = "y"     'used as flag to reenter parameter.

END SELECT

SELECT CASE S2method%    'find S2 associated with
                          hypothesized values.
      CASE 1
          FOR i = 1 to numPoints%
             S2hypX(i) = propConstXS2 * (hypX(i) ^ expXS2)
             S2hypY(i) = propConstYS2 * (hypY(i) ^ expYS2)
          NEXT
      CASE 2
          FOR i = 1 to numPoints%
             S2hypX(i) = S2X(i)
```

```
            S2hypY(i) = S2Y(i)
         NEXT
END SELECT

CALL Print2RealVariables ( " logHypX        ", " logHypY",
  logHypX(), logHypY(), numPoints%)
CALL Print2RealVariables ( "  hypX          ", "  hypY",
  hypX(), hypY(), numPoints%)
CALL Print2RealVariables ( " S2hypX         ", " S2hypY",
  S2hypX(), S2hypY(), numPoints%)
Input "Error check. Continue (y/n)"; continue$
cls
IF continue$ = "n" OR reTry$ = "y" THEN
   PRINT "An error was made in entering a parameter, please
     try again."
   PRINT " "
   PRINT " "
   CALL HypothesizedXY (logHypX(), hypX(), logHypY(),
     hypY(), S2hypX(), S2hypY())
END IF
END SUB


SUB Print2RealVariables  ( var1$, var2$, var1(1), var2(1),
  numPoints%)
REM prints a 2 variable table of values. (var1 starts at
  10th spot, then 6 blanks)
PRINT "lake     "; var1$; var2$
PRINT "----     ----------------------------------"

FOR i = 1 to numPoints%
   a$ = "###      ######.####      ######.####"
   PRINT USING a$; i, var1(i), var2(i)
   IF Fix(i/12) - (i/12) = 0  THEN     'stops screan output
                                        after 12 values.
      PPINT " "
      PRINT " "
      INPUT "press any key to continue";carryOn$
   END IF
NEXT

PRINT " "
INPUT "would you like this data filed (y/n)";pleaseFile$
IF pleaseFile$ = "y"  THEN
   PRINT " "
   INPUT "what file name would you like? (DOS
     format)";fileName$
   CALL CreateFile (fileName$, var1(), var2(), numPoints%)
END IF
cls
END SUB
```

```
SUB MinMaxCICheck (meanXorY$, Q(1), Qmin(1), Qmax(1),
  hypQ(1), sampleSizeQ%(1), S2Q(1), S2hypQ(1))
REM: Checks if preset max or min are below or above
  hypothesized X or Y values.
REM: Also checks if CI of real data X and Y include
  hypothesized X and Y.
SHARED boundsType%, repBoundsType%, numPoints%, lowerCI(),
  upperCI()  'lower and upperCI() are shared only to
             dimension arrays
LOCAL i, max, min, carryOn$, maxQbelowHypQ%, minQaboveHypQ%,
  flag$, flagIt$

maxQbelowHypQ% = 0     'initializes varible to count when
                        preset lower bound on meanY is above
                        real data regression line.
minQaboveHypQ% = 0

FOR i = 1 to numPoints%
  IF (Qmin(i) > hypQ(i)) THEN  'check for min over
                               hypothesized value.
     minQaboveHypQ% = minQaboveHypQ% + 1
  END IF

  IF boundsType% = 1 THEN
       'skip any investigation of max (because it is  not
        invoked in type 1)
  ELSE
    IF (Qmax(i) < hypQ(i))  THEN  'check for max under
                                  hypothesized value.
       maxQbelowHypQ% = maxQbelowHypQ% + 1
    END IF
  END IF
NEXT

PRINT " "
PRINT " "
PRINT "TALLY : based on ";numPoints%;" fitted "; meanXorY$;"
  values."
PRINT
"-----------------------------------------------------"
PRINT "preset min ";meanXorY$;" is above hypothesized value
  ";minQaboveHypQ%;" times."

IF boundsType% = 1 THEN
   'skip the printing of any max meanQ info
ELSE
   PRINT "preset max ";meanXorY$;" is below hypthosized
     value ";maxQbelowHypQ%;" times. "
END IF

PRINT " "
```

```
PRINT " "
PRINT " a non-zero value suggests that random error due to
  sampling is unlikely "
PRINT "  to account for some of real data,  even if model
  SSE excedes real SSE."
PRINT " "
INPUT "press any key to continue"; carryOn$
CLS



CALL FindConfidenceIntervals (Q(), numPoints%,
  sampleSizeQ%(), S2Q(), lowerCI(), upperCI())

PRINT "                  ";meanXorY$;" : measured vs.
  hypothesized values."
PRINT "
----------------------------------------"
PRINT "When hypothesized value is outside CI for data value,
  sampling error "
PRINT "is unlikely to account for the gap between the
  hypothesized and data value."
PRINT " "
PRINT " "
PRINT "Lake    (lower95CI - data value - upper95CI)
  hyp. value      "
PRINT
"------------------------------------------------------------
-------------"
PRINT " "

FOR i = 1 to numPoints%
  IF (hypQ(i) > lowerCI(i)) AND (hypQ(i) < upperCI(i)) THEN
     flagIt$ = "OK"
  ELSEIF hypQ(i) <= lowerCI(i) THEN
     flagIt$ = "**LoHyp**"
  ELSEIF hypQ(i) >= upperCI(i) THEN
     flagIt$ = "**HiHyp**"
  END IF

  PRINT USING "###      &####.### & ####.### & ####.###&
      ####.###
&";i,"(",lowerCI(i),"-",Q(i),"-",upperCI(i),")",hypQ(i),flag
It$
  IF INT(i/15) - (i/15) = 0 THEN
    INPUT "More, press any key";carryOn$
    PRINT " "
  END IF
NEXT i
PRINT " "
INPUT "press any key to continue";carryOn$
cls
```

```
CALL FindConfidenceIntervals (hypQ(), numPoints%,
  sampleSizeQ%(), S2hypQ(), lowerCI(), upperCI())
SELECT CASE boundsType%
  CASE 1
     PRINT "                    ";meanXorY$;" : bounds for
       simulation."
     PRINT "                    ------------------------------"
     PRINT " This table helps to spot if hypothesized value
       is below  min, "
     PRINT " and which of min, max, lowerCI or upperCI will
       bind simulations."
     PRINT " "
     PRINT " "
     PRINT "Lake      min            lowerCI - hyp.value -
       upperCI       "
     PRINT
"-------------------------------------------------------------
----------------"
     a$ =  "###    ####.####      #####.#### ! ####.#### !
       ####.####       & "

     FOR i = 1 to numPoints%
        IF (Qmin(i) < lowerCI(i)) THEN
           flag$ = "OK"
        ELSEIF (Qmin(i) > hypQ(i)) THEN
           flag$ = "***LoHyp***"
        ELSE
           flag$ = "***CIflag***"
        END IF

        PRINT USING a$; i, Qmin(i), lowerCI(i), "-",
          hypQ(i), "-", upperCI(i), flag$
        IF INT(i/15) - (i/15) = 0 THEN
            INPUT "More, press any key";carryOn$
        PRINT " "
        END IF
     NEXT

  CASE 2, 3
     PRINT "                    ";meanXorY$;" : bounds for
       simulation."
     PRINT "                    ------------------------------"
     PRINT " This table helps to spot if hypothesized value
       is below min or over max, "
     PRINT " and which of min, max, lowerCI or upperCI will
       bind simulations."
     PRINT " "
     PRINT " "
     PRINT "Lake         min - max              lowerCI -
       hyp.value - upperCI           "
     PRINT
```

```
"------------------------------------------------------------
------------------"
     a$ =  "###    ####.#### ! ####.####     #####.#### !
        ####.#### ! ####.####   &"

     FOR i = 1 to numPoints%
        IF (Qmin(i) < lowerCI(i)) AND (Qmax(i) > upperCI(i))
          THEN
           flag$ = "OK"
        ELSEIF (Qmin(i) >= hypQ(i)) THEN
           flag$ = "**LoHyp**"
        ELSEIF (Qmax(i) <= hypQ(i)) THEN
           flag$ = "**HiHyp**"
        ELSE
          flag$ = "**CIflag**"
        END IF

        PRINT USING a$; i, Qmin(i), "-", Qmax(i),
          lowerCI(i), "-", hypQ(i), "-", upperCI(i), flag$
        IF INT(i/15) - (i/15) = 0 THEN
            INPUT "More, press any key";carryOn$
        PRINT " "
        END IF
     NEXT
END SELECT

INPUT "press any key to continue";continue$
cls
END SUB


SUB FindConfidenceIntervals (W(1), numPoints%,
  sampleSizeW%(1), S2W(1), lowerCI(1), upperCI(1))
REM finds confidence intervals for each mean W using
   mean:variance or real data S2.
LOCAL S2, i, SE
     FOR i = 1 to numPoints%
       S2 = S2W(i)
       SE = SQR ( S2 / sampleSizeW%(i) )
       upperCI(i) = W(i) + 1.96 * SE
       lowerCI(i) = W(i) - 1.96 * SE
     NEXT i
END SUB


SUB FindRepConfidenceIntervals (W(1), numPoints%, S2Method%,
  expWS2, propConstWS2, S2W(1), lowerCI(1), upperCI(1))
REM finds confidence intervals for replicat W using
  mean:variance or real data S2.
SELECT CASE S2method%
```

```
  CASE 1
     FOR i = 1 to numPoints%
       S2 = propConstWS2 * W(i) ^ expWS2
       SD = SQR(S2)
       upperCI(i) = W(i) + 1.96 * SD
       lowerCI(i) = W(i) - 1.96 * SD
     NEXT i
  CASE 2
     FOR i = 1 to numPoints%
       S2 = S2W(i)
       SD = SQR(S2)
       upperCI(i) = W(i) + 1.96 * SD
       lowerCI(i) = W(i) - 1.96 * SD
     NEXT i
END SELECT
END SUB


SUB RepXminMaxCIcheck (numPoints%, X(1), boundsType%,
  minRepx, maxRepx, Xmin(1), Xmax(1))
REM Creates a table for meanX and repx CI, min and max.
  Shows whether CI or min, max bind simulations.
  $INCLUDE  "repXCI.inc"  '*** not in this appendix ***
END SUB


SUB SimulateSamples (hypY(1), sampleSizeX%(1),
  sampleSizeY%(1), numPoints%, expXS2, propConstXS2, expYS2,
  propConstYS2, hypX(1), logRandX(1), logRandy(1))
REM Randomizes mean X and perfectFit Y (both
  non-transformed), then log-
REM transforms them.  Randomizations mimic sampling from a
  Normal dist. with
REM hypX(i) or hypY(i) as mean, and an SD derived from an
  appropriate
REM S2:M regression or from data S2.
LOCAL i, meanXupperCI, meanXlowerCI, meanYupperCI,
  meanYlowerCI, simulatedMeanX, simulatedMeanY
SHARED boundsType%, meanXboundByHi95CI&,
  meanXboundByLo95CI&, meanXboundByMin&, meanXboundByMax&,
  minMeanXisUnder95CI&, maxMeanXisOver95CI&, Xmax(),
  Xmin(),_
  meanYboundByHi95CI&, meanYboundByLo95CI&,
  meanYboundByMin&, meanYboundByMax&, minMeanYisUnder95CI&,
  maxMeanYisOver95CI&, Ymax(), Ymin()
SHARED S2method%, S2hypX(), S2hypY(), simulationsForX&,
  simulationsForY&

FOR i = 1 to numPoints%
   CALL Randomization ((S2hypX(i)), sampleSizeX%(i),
```

```
      hypX(i), simulatedMeanX, meanXupperCI, meanXlowerCI)
    INCR simulationsForX&          'counts simulations
    CALL ApplyBounds (boundsType%, simulatedMeanX,
      meanXupperCI, meanXlowerCI, Xmax(i), Xmin(i),
      meanXboundByHi95CI&, meanXboundByLo95CI&,
      meanXboundByMax&, meanXboundByMin&,
      maxMeanXisOver95CI&, minMeanXisUnder95CI&)
    logRandx(i) = LOG10 (simulatedMeanX)

    CALL Randomization ((S2hypY(i)), sampleSizeY%(i),
      hypY(i), simulatedMeanY, meanYupperCI, meanYlowerCI)
    INCR simulationsForY&
    CALL ApplyBounds (boundsType%, simulatedMeanY,
      meanYupperCI, meanYlowerCI, Ymax(i), Ymin(i),
      meanYboundByHi95CI&, meanYboundByLo95CI&,
      meanYboundByMax&, meanYboundByMin&,
      maxMeanYisOver95CI&, minMeanYisUnder95CI&)
    logRandy(i) = LOG10 (simulatedMeanY)
NEXT i
END SUB



SUB SimulateWhenReplicatesCorr ( sampleSizeX%(1),
  sampleSizeY%(1), numPoints%, meanX(1), expXS2,
  propConstXS2, hypY(1), expYS2, propConstYS2,
  repLevelSlope(1), repLevelIntercept(1),
  repLevelS2YgivenX(1), logRandX(1), logRandY(1))
REM simulates sampleSizeX replictes of meanX, for each of
  these a replicate y is simulated using the rep. level
  regression between rep.x and rep. y, and S2y|x (rep.
  level).
REM note that  xMax(), xMin(), yMax(), yMin() are used as
  bounds for both means and replicates, if boundsType% 3 and
  repBoundsType% 3 are chosen.
LOCAL i, j, sumX, sumY, simReplicateX, simReplicateY,
  simulatedMeanX, simulatedMeanY, carryOn$, errorCondition%,
  S2forX, S2forY, SEmeanX, SEmeanY
SHARED boundsType%, meanXboundByHi95CI&,
  meanXboundByLo95CI&, meanXboundByMin&, meanXboundByMax&,
  minMeanXisUnder95CI&, maxMeanXisOver95CI&, Xmax(),
  Xmin(),_
  meanYboundByHi95CI&, meanYboundByLo95CI&,
  meanYboundByMin&, meanYboundByMax&, minMeanYisUnder95CI&,
  maxMeanYisOver95CI&, Ymax(), Ymin()
SHARED repBoundsType%, repxBoundByHi95CI&,
  repxBoundByLo95CI&, repxBoundByMin&, repxBoundByMax&,
  minRepxIsUnder95CI&, maxRepxIsOver95CI&, minRepx,
  maxRepx,_
  repyBoundByHi95CI&, repyBoundByLo95CI&, repyBoundByMin&,
  repyBoundByMax&, minRepyIsUnder95CI&, maxRepyIsOver95CI&,
```

```
  minRepy, maxRepy
SHARED S2method%, S2X(), S2Y(), simulationsForX&,
  simulationsForY&

FOR i=1 to numPoints%

   SELECT CASE S2method%
     CASE 1
        S2forX = (propConstXS2) * (meanX(i) ^ expXS2)
          'from a logS2:logX plot.
        S2forY = (propConstYS2) * (hypY(i) ^ expYS2)
     CASE 2
        S2forX = S2X(i)
        S2forY = S2Y(i)
   END SELECT

   SEmeanX = SQR (S2forX / sampleSizeX%(i))
   SEmeanY = SQR (S2forY / sampleSizeY%(i))

   meanXupper95CI = meanX(i) + 1.96 * SEmeanX
   meanXlower95CI = meanX(i) - 1.96 * SEmeanX

   meanYupper95CI = hyp(i) + 1.96 * SEmeanY
   meanYlower95CI = hypY(i) - 1.96 * SEmeanY

   counter% = 0
   sumx = 0
   sumy = 0

   Do
     counter% = counter% + 1
     CALL RandomizeReplicateX (i, meanX(i), S2forX,
       simulatedRepx)
     sumx = sumx + simulatedRepx
     CALL RandomizeReplicateY (i, hypY(i), repLevelSlope(i),
       simulatedRepx, repLevelIntercept(i),
       repLevelS2YgivenX(i), simulatedRepy)
       sumy = sumy + simulatedRepy
   Loop until (counter% = sampleSizeX%(i)) OR (counter% =
     sampleSizeY%(i))

   IF sampleSizeX%(i) > sampleSizeY%(i) THEN
      DO
        counter% = counter% + 1
        CALL RandomizeReplicateX (i, meanX(i), S2forX,
          simulatedRepx)
        sumx = sumx + simulatedRepx
      Loop until counter% = sampleSizeX%(i)
   ELSE
     IF sampleSizeY%(i) > sampleSizeX%(i) THEN
          'and if samplesizes are equal we are finished.
        DO
```

```
            counter% = counter% + 1
            CALL RandomizeReplicateX (i, meanX(i), S2forX,
              simulatedRepx)
            CALL RandomizeReplicateY (i, hypY(i),
              repLevelSlope(i), simulatedRepx,
              repLevelIntercept(i), repLevelS2YgivenX(i),
              simulatedRepy)
            sumy = sumy + simulatedRepy
          Loop until counter% = sampleSizeY%(i)
        END IF
      END IF

      simulatedMeanX = (sumx/sampleSizeX%(i))
      simulatedMeanY = (sumy/sampleSizeY%(i))

      CALL ApplyBounds (boundsType%, simulatedMeanX,
        meanXupper95CI, meanXlower95CI, Xmax(i), Xmin(i),
        meanXboundByHi95CI&, meanXboundByLo95CI&,
        meanXboundByMax&, meanXboundByMin&,
        maxMeanXisOver95CI&, minMeanXisUnder95CI&)
      logRandx(i) = LOG10 (simulatedMeanX)

      CALL ApplyBounds (boundsType%, simulatedMeanY,
        meanYupper95CI, meanYlower95CI, Ymax(i), Ymin(i),
        meanYboundByHi95CI&, meanYboundByLo95CI&,
        meanYboundByMax&, meanYboundByMin&,
        maxMeanYisOver95CI&, minMeanYisUnder95CI&)
      logRandy(i) = LOG10 (simulatedMeanY)
NEXT i
END SUB


SUB Graphics (boundsType%, Xmin(1), Xmax(1), Ymin(1),
  Ymax(1), numPoints%, logRealX(1), logRealY(1), realSlope,
  realIntercept, logRandX(1), logRandY(1), slope ,
  intercept)
REM graphs real data and regression in upper part of screen
  and simulated data and
REM regression in lower part of screen.
LOCAL maxLogX, minLogX, maxLogY, minLogY, initialFittedLogY,
  finalFittedLogY, carryOn$
SHARED modelSSE, realSSE, sortedlogRealX(),
  sortedlogRealY(), sortedlogRandX(), sortedXmin(),
  sortedXmax(), sortedYmin(), sortedYmax()


FOR i = 1 to numPoints%     'preserves original arrays
   sortedlogRealX(i) = logRealX(i)     'some sorts not new
                                        for each CALL,
   sortedLogRealY(i) = logRealY(i)     'but are here for
                                        clarity.
```

```
   sortedlogRandX(i) = logRandX(i)
   sortedXmax(i) = Xmax(i)
   sortedYmax(i) = Ymax(i)
   sortedXmin(i) = Xmin(i)
   sortedYmin(i) = Ymin(i)
NEXT
CALL Quicksort (1,(numPoints%), sortedlogRealX())
  'used for screen sizing
CALL Quicksort (1,(numPoints%), sortedlogRealY())
CALL Quicksort (1,(numPoints%), sortedlogRandX())
CALL Quicksort (1,(numPoints%), sortedXmax())
CALL Quicksort (1,(numPoints%), sortedYmax())
CALL Quicksort (1,(numPoints%), sortedXmin())
CALL Quicksort (1,(numPoints%), sortedYmin())

cls
screen 1,0
'graphics during anultest?

IF boundsType% = 1 THEN        ' ie. no max specified
   maxLogX = 1.4 * sortedlogRealX(numPoints%)
                          ' scales viewport to data
   maxLogY = 1.4 * sortedlogRealY(numPoints%)
ELSE
   maxLogX = LOG10 (sortedXmax(numPoints%))
   maxLogY = LOG10 (sortedYmax(numPoints%))
END IF

minLogX = LOG10(sortedXmin(1))  'LOG10(.05)  this will
                                 shrink line when min is low
minLogY = LOG10(sortedYmin(1))  'LOG10(.05)
window (minLogX, minLogY) - (maxLogX, maxLogY)



view (25,0) - (319, 75)      'upper part of screen prepared
                              for real data

initialFittedLogY = realSlope * sortedLogRealX(1) +
realIntercept                              'may require sort****
finalFittedLogY = realSlope * sortedLogRealX(numPoints%) +
realIntercept

line (sortedLogRealX(1), initialFittedLogY) -
(sortedLogRealX(numPoints%), finalFittedLogY)    'real data
                                                  regression

FOR i = 1 to numPoints%                  'real data points
  PSET (logRealX(i), logRealY(i)), 14
NEXT
```

```
view (25,100) - (319,175)     'lower part of screen prepared
                               for simulated data

initialFittedLogY = slope * sortedLogRandX(1) + intercept
finalFittedLogY = slope * sortedLogRandX(numPoints%) +
intercept

line (sortedLogRandX(1), initialFittedLogY) -
(sortedLogRandX(numPoints%), finalFittedLogY)   'simulated
                                                 data
                                                 regression

FOR i = 1 to numPoints%
   PSET (logRandX(i), logRandY(i)), 10
NEXT


LOCATE 1,1
PRINT "real SSE= "
PRINT USING "##.####";realSSE
LOCATE 15,1
PRINT "model SSE = "
PRINT USING "##.####";modelSSE
LOCATE 25,2
INPUT "press";carryOn$
cls
cls
screen 0
width 80
END SUB



SUB CompileSSEdist (modelSSE, modelSSY, realSSE, realSSY,
  numPoints%, trial, unexplainedMSE(1), modelr2(1),
  pseudor2(1), arrayModelSSE(1), modelSSEoverSSY(1),
  arrayModelSSY(1) )
REM builds arrays of modelSSE etc.
SHARED  wantSSEoverSSYfile$, wantSSYfile$
LOCAL unexplainedSSE, carryOn$
   unexplainedSSE = realSSE - modelSSE
   IF unexplainedSSE < 0 THEN unexplainedSSE = 0
   unexplainedMSE(trial) = unexplainedSSE / numPoints%
   modelr2(trial) = (modelSSY - modelSSE) / modelSSY
   pseudor2(trial) = (realSSY - unexplainedSSE) / realSSY
   arrayModelSSE(trial) = modelSSE
   IF wantSSEoverSSYfile$ = "y" THEN  modelSSEoverSSY(trial)
     = modelSSE / modelSSY
   IF wantSSYfile$ = "y"        THEN  arrayModelSSY(trial)
     = modelSSY
   'print "arrayModelSSE(trial) = ";arrayModelSSE(trial)
```

```
      ***used for checks***
    'print "arraymodelSSY(trial)  =";arrayModelSSY(trial)
    'print "modelSSEoverSSY(trial)="; modelSSEoverSSY(trial)
    'print "model r2=" modelr2(trial)
    'input "press";c$
END SUB


SUB ComputeSSEstats ( unexplainedMSE(1), modelr2(1),
  pseudor2(1), numTrials%, meanUnexplainedMSE, meanModelr2,
  meanPseudor2, arrayModelSSE(1), realSSE,
  probModelSSEgtRealSSE, meanModelSSE, lowerBoundMeanSSE,
  upperBoundMeanSSE )
REM takes arrays of simulation results: unexplainedMSE(),
  model r2(), pseudor2(), and arrayModelSSE() -
REM determines means, then sorts arrays into ascending order
  and finds 95% CI.

SHARED CIwarning$, lowerBoundMSE, upperBoundMSE,
  lowerBoundModelr2, upperBoundModelr2, lowerBoundPr2,
  upperBoundPr2, wantSSEfile$, modelSSEoverSSY(),
  wantSSEoverSSYfile$, wantSSYfile$, arrayModelSSY(),
  wantR2file$
LOCAL trial, sumUnexplainedMSE, sumPseudor2, sumModelSSE,
  pos2.5%, pos97.5%

   sumUnexplainedMSE = 0
   sumModelr2 = 0
   sumPseudor2 = 0
   sumModelSSE = 0

   FOR trial = 1 to numTrials%
     sumUnexplainedMSE = unexplainedMSE(trial) +
       sumUnexplainedMSE
     sumModelr2 = modelr2(trial) + sumModelr2
     sumPseudor2 = pseudor2(trial) + sumPseudor2
     sumModelSSE = arrayModelSSE(trial) + sumModelSSE
   NEXT trial

   meanUnexplainedMSE = sumUnexplainedMSE / numTrials%
   meanModelr2 = sumModelr2 / numTrials%
   meanPseudor2 = sumPseudor2 / numTrials%
   meanModelSSE = sumModelSSE / numTrials%

   CALL QuickSort (1,(numTrials%), unexplainedMSE())
     'a sort procedure
   CALL QuickSort (1,(numTrials%), pseudor2())
   CALL QuickSort (1,(numTrials%), modelr2())
   CALL QuickSort (1,(numTrials%), arrayModelSSE())

   IF numTrials% < 50 THEN
     pos2.5% = 1
```

```
    pos97.5% = numTrials% - 1
    CIwarning$ = "y"          'a small sample size warning
  ELSE
    pos2.5% = INT ( .025 * numTrials% )
    pos97.5% = INT ( .975 * numTrials% )
  END IF

  lowerBoundMSE = unexplainedMSE( pos2.5%)
  upperBoundMSE = unexplainedMSE( pos97.5%)
  lowerBoundModelr2 = modelr2( pos2.5%)
  upperBoundmodelr2 = modelr2( pos97.5%)
  lowerBoundPR2 = pseudor2( pos2.5%)
  upperBoundPR2 = pseudor2( pos97.5%)
  lowerBoundMeanSSE = arrayModelSSE( pos2.5%)
  upperBoundMeanSSE = arrayModelSSE( pos97.5%)

  IF wantSSEfile$ = "y" THEN
    INPUT "What filename would you like for modelSSE array
      "; moniker$
    CALL Create1RealVariableFile (moniker$, numTrials%,
      arrayModelSSE())
  END IF
  IF wantSSEoverSSYfile$ = "y" THEN
    INPUT "What filename would you like for modelSSE/SSY
      array ";moniker$
    CALL Create1RealVariableFile (moniker$, numTrials%,
      modelSSEoverSSY())
  END IF
  IF wantSSYfile$ = "y" THEN
    INPUT "What filename would you like for modelSSY array
      ";moniker$
    CALL Create1RealVariableFile (moniker$, numTrials%,
      arrayModelSSY())
  END IF
  IF wantR2file$ = "y" THEN
    INPUT "What filename would you like for modelr2 array
      ";moniker$
    CALL Create1RealVariableFile (moniker$, numTrials%,
      modelr2())
  END IF
END SUB


SUB Create1RealVariableFile (moniker$, numPoints%, a(1))
REM creates a 1 real variable external data file.
LOCAL  count%, carryOn$

OPEN moniker$ FOR OUTPUT AS #1
count% = 0
DO
  count% = count% + 1
  PRINT #1, a(count%)
```

```
LOOP UNTIL count% = numPoints%
CLOSE #1
PRINT "File "; moniker$;" created."
INPUT "press any key to continue";carryOn$
cls
END SUB


$INCLUDE "Boundspr.inc"    'has two procedures for printing
  boundsinfringemnts.


SUB QuickSort ( start%, finish%, datum(1) )  'ascending sort
REM Recursively sorts array called datum, with bounds start%
  and finish%.
REM This procedure is from a book called Oh!Pascal (Cooper
  and Clancy, 1985) and was translated to
REM TurboBasic. A sort is needed to find 95% CI's for
  simulated statistics.
LOCAL left%, right%, starterValue, temp
  left%=start%
  right%=finish%
  starterValue=datum((start%+finish%)\2)
  DO
    DO WHILE datum(left%)<starterValue
       left%=left%+1
    LOOP
    DO WHILE starterValue < datum(right%)
       right%=right%-1
    LOOP
    IF left% <= right% THEN
       temp = datum(left%)
       datum(left%) = datum(right%)
       datum(right%) = temp
       left% = left% + 1
       right% = right% - 1
    END IF
  LOOP UNTIL right% <= left%
  IF start% < right% THEN CALL QuickSort ((start%),
    (right%), datum())
  IF left% < finish% THEN CALL QuickSort ((left%),
    (finish%), datum())
END SUB


SUB FindProbModelSSEgtRealSSE ( realSSE, numTrials%,
  arrayModelSSE(1), probModelSSEgtRealSSE )
REM finds the number of times modelSSE matched or exceeded
  realSSE, ie
REM 'explained' all scatter around real-data regression
  line.
LOCAL count%
```

```
  count% = numTrials%
  DO UNTIL   (count% = 0)  OR  (realSSE >=
     arrayModelSSE(count%))
    PRINT "in FindProbRealSSE DO LOOP "; count%
    count% = count% - 1
  LOOP
  probModelSSEgtRealSSE = (numTrials% - count%) / numTrials%
END SUB


SUB Randomization (S2, sampleSize%, meanValue,
  simulatedValue, upper95CI, lower95CI)
REM  Randomly samples a value from a Normal (meanValue,
  SQR(S2/n)) dist.
REM  Note that it is Non-transformed values of x and y that
  are manipulated.
LOCAL SE, rand, varMean
SHARED S2method%

rand = RND                    'uniform distribution.

varMean = S2 / sampleSize%     'when not a mean (ie.
                               sampleSize = 1) S2 is not
                               altered.
SE = SQR ( varMean )
simulatedValue = (.5513 * LOG( rand / (1-rand) )) * SE +
  meanValue    'normal dist.
upper95CI = meanValue + (1.96 * SE)
lower95CI = meanValue - (1.96 * SE)
END SUB


SUB RandomizeReplicateX (i, meanX, S2, simulatedRepx)
REM  Randomly samples a value from a Normal (meanValue,
  SQR(S2/n)) dist.
REM  and truncates distribution to fit specified bounds.
REM  Note that it is Non-transformed values of x and y that
  are manipulated.
LOCAL rand, SD
SHARED repBoundsType%, repxBoundByHi95CI&,
  repxBoundByLo95CI&, repxBoundByMin&, repxBoundByMax,
  minRepxIsUnder95CI&, maxRepxIsOver95CI&, minRepx, maxRepx,
  xMax(), xMin(), simulationsForX&


   rand = RND              'uniform distribution.
   SD = SQR ( S2 )

   simulatedRepx = (.5513 * LOG( rand / (1-rand) )) * SD +
     meanX    'normal dist. with SD of disaggregated data.
```

```
    INCR simulationsForX&
    ' PRINT "in randomizeRX, before bounds ,  sim rep x =
       ";simulatedRepx   '***

    repxUpperCI = meanX + 1.96 * SD
    repxLowerCI = meanX - 1.96 * SD

    IF repBoundsType% = 3 THEN
       maxRepx = xMax(i)     'sets max and min specific for
                              the ith lake.
       minRepx = xMin(i)
    END IF

    CALL ApplyBounds (repBoundsType%, simulatedRepx,
      repxUpperCI, repxLowerCI, maxRepx, minRepx,
      repxBoundByHi95CI&, repxBoundByLo95CI&,
      repxBoundByMax&, repxBoundByMin&, maxRepxIsOver95CI&,
      minRepxIsUnder95CI&)
      ' PRINT "in randomizeRX, after bounds ,  sim rep x =
         ";simulatedRepx   '***
END SUB


SUB RandomizeReplicateY (i, perfectFittingMeanY, slope, x,
  intercept, S2repyGivenRepx, simulatedRepy)
REM A function that simulates y values at the disaggregated
  level -
REM it takes a simulated x value
REM and then simulates an appropriatly constrained y value
  by using the x:y
REM regression at disaggregated level, and S2YgivenX at that
  level. Note that
REM most published bottom-up regressions use aggregated data
  points.

LOCAL rand, y, SD
SHARED repBoundsType%, repyBoundByHi95CI&,
  repyBoundByLo95CI&, repyBoundByMax&, repyBoundByMin&,
  minRepyIsUnder95CI&, maxRepyIsOver95CI&, minRepy, maxRepy,
  yMax(), yMin(), simulationsForY&

    rand = RND                          'uniform distribution.

    y = slope * x + intercept     'x is a simulated replicate,
                                   y is on a line through
                                   replicate data.
    SD = SQR (S2repyGivenRepx)    'S2y|x from regression
                                   through replicates.

    simulatedRepy = (.5513 * LOG( rand / (1-rand) )) * SD + y
      'normal dist. with SD of disaggregated data.
    INCR simulationsForY&
```

```
    repyUpperCI = y + 1.96 * SD
    repyLowerCI = y - 1.96 * SD

    IF repBoundsType% = 3 THEN
       maxRepy = yMax(i)     'sets max and min specific for
                              the ith lake.
       minRepy = yMin(i)
    END IF

    CALL ApplyBounds (repBoundsType%, simulatedRepy,
      repyUpperCI, repyLowerCI, maxRepy, minRepy,
      repyBoundByHi95CI&, repyBoundByLo95CI&,
      repyBoundByMax&, repyBoundByMin&, maxRepyIsOver95CI&,
      minRepyIsUnder95CI&)
END SUB


SUB  ApplyBounds (limits%, simulatedValue, upperCI, lowerCI,
  max, min, boundByHiCI&, boundByLoCI&, boundByMax&,
  boundByMin&, maxOverHiCI&, minUnderLoCI&)
REM  places simulated mean or replicate values within
  prescribed bounds.

   Select Case limits%       'fits simulated value to preset
                              bounds

         Case 0              'no bounds on simulated values.
                              (for future program)

         Case 1              'bound by Confidence Interval and
                              an arbitrary minimum.

                              '___upper bound___

           IF simulatedValue > upperCI THEN
              boundByHiCI& = boundByHiCI& + 1
              simulatedValue = upperCI
           END IF

                             '___lower bound___

           IF min < lowerCI THEN    'higher of the two will
                                     be bounds
              minUnderLoCI& = minUnderLoCI& + 1
                'tells us about min wrt. CI

              IF simulatedValue < lowerCI THEN
                'if > lowerCI then no need for bounding.

                 simulatedValue = lowerCI
                 boundByLoCI& = boundByLoCI& + 1
```

```
      END IF

   ELSE          'ie. lowerCI <= min
      IF simulatedValue < min THEN
         simulatedValue = min
         boundByMin& = boundByMin& + 1
      END IF
   END IF


Case 2,3     'bound by 95%CI and a max and min value
              (specific to each lake in case 3)

                    '___upper bound___

   IF max > upperCI THEN    'lower of the two will be
                             bounds
      maxOverHiCI& = maxOverHiCI& + 1
        'tells us about min wrt. CI

      IF simulatedValue > upperCI THEN
         simulatedValue = upperCI
         boundByHiCI& = boundByHiCI& + 1
      END IF

   ELSE        'ie. upperCI >= max
      IF simulatedValue > max THEN
         simulatedValue = max
         boundByMax& = BoundByMax& + 1
      END IF
   END IF

                   '___lower bound___

   IF min < lowerCI THEN  'higher of the two will be
                          bounds
      minUnderLoCI& = minUnderLoCI& + 1
        'tells us about min wrt. CI

      IF simulatedValue < lowerCI THEN
         simulatedValue = lowerCI
         boundByLoCI& = boundByLoCI& + 1
      END IF

   ELSE       'ie. lowerCI <= min
      IF simulatedValue < min THEN
         simulatedValue = min
         boundByMin& = boundByMin& + 1
      END IF
   END IF

END SELECT
```

```
END SUB


SUB SoundAlert (nullVariable)
REM notifies user when input required
              FOR n = 1 to 5
                SOUND 500, .01
                DELAY .4
                SOUND 2000, .01
                DELAY .04
              NEXT n

              FOR n = 1000 to 700 step  -5
                SOUND n,1
              NEXT n

              FOR n = 1 to 700
                SOUND 50 * RND + 37, .0015
              NEXT n
END SUB



*** because of memory restrictions, the following pocedure
was accessed through the $INCLUDE statement ***

SUB PrintBoundsInfringements ( numPoints%, numTrials%,
  boundsType%, meanXboundByMin&, meanXboundbyMax&,
  meanXboundByHi95CI&, meanXboundByLo95CI&,
  minMeanXisUnder95CI&, maxMeanXisOver95CI&,_

  meanYboundByMin&, meanYboundByMax&, meanYboundByHi95CI&,
  meanYboundByLo95CI&, minMeanYisUnder95CI&,
  maxMeanYisOver95CI&)
REM prints the number of times that bounds for simulated
  meanX or meanY are crossed.
LOCAL carryOn$, sumNX%, sumNY%, totalMeanX&, totalMeanY&
SHARED simulationsForX&, simulationsForY&, sampleSizeX%(),
  sampleSizeY%(), replicateCorrelation$, loopCounter&
'**** below not needed if loopCounter& is used
'IF replicateCorrelation$ = "y" THEN  'IF block
                                       needed for tally
'    sumNX% = 0
'    sumNY% = 0
'    FOR i = 1 to numPoints%
'        sumNX% = sampleSizeX%(i) + sumNX%
      'sum of sampleSize from each location
'        sumNY% = sampleSizeY%(i) + sumNY%
'    NEXT i
'    totalMeanX& = (simulationsForX& / sumNX%) * numPoints%
```

```
'    totalMeanY& = (simulationsForY& / sumNY%) * numPoints%
'ELSE
'    totalMeanX& = simulationsForX&
'    totalMeanY& = simulationsForY&
'END IF

totalMeanX& = loopCounter& * numPoints% *  numTrials%
totalMeanY& = totalMeanX&

Select Case boundsType%
CASE 1
 PRINT " "
 PRINT " Tally: ";  totalMeanX&;" simulations of mean X / ";
   totalMeanY&;" simulations for mean Y"
 PRINT
"-----------------------------------------------------------
-----------------"
 PRINT " "
 PRINT "initial simulated meanX was bound by  min: (";
   meanXboundByMin&;") times."
 PRINT "intiial simulated meanX was bound by (Hi/Lo) 95% CI:
   ("; meanXboundByHi95CI& ;"/"; meanXboundByLo95CI&;")
     times."
 PRINT "arbitrary min meanX is under 95% CI:
   (";minMeanXisUnder95CI&;") times."
 PRINT " "

 PRINT "initial simulated meanY was bound by min: (";
   meanYboundByMin&;") times. "
 PRINT "initial simulated meanY was bound by (Hi/Lo) 95% CI:
   ("; meanYboundByHi95CI&;"/"; meanYboundByLo95CI&;")
     times."
 PRINT "arbitrary min meanY is under 95% CI:
   (";minMeanYisUnder95CI&;") times."
 PRINT " "

CASE 2,3
 PRINT " "
 PRINT " Tally: ";  totalMeanX&;" simulations of mean X / ";
   totalMeanY&;" simulations for mean Y"
 PRINT
"-----------------------------------------------------------
-----------------"
 PRINT " "
 PRINT "initial simulated meanX was bound by (max/min): (";
   meanXboundByMax&;"/";meanXboundByMin&;") times."
 PRINT "intiial simulated meanX was bound by (Hi/Lo) 95% CI:
   ("; meanXboundByHi95CI& ;"/"; meanXboundByLo95CI&;")
     times."
 PRINT "max meanX and min meanX are (over/under) 95% CI:
   (";maxMeanXisOver95CI&;"/";minMeanXisUnder95CI&;")
     times."
```

```
 PRINT " "

 PRINT "initial simulated meanY was bound by (max/ min): (";
   meanYboundByMax&;"/";meanYboundbyMin&;") times. "
 PRINT "initial simulated meanY was bound by (Hi/Lo) 95% CI:
   ("; meanYboundByHi95CI&;"/"; meanYboundByLo95CI&;")
     times."
 PRINT "max meanY and min meanY are (over/under) 95% CI:
   (";maxMeanYisOver95CI&;"/";minMeanYisUnder95CI&;")
     times."
 PRINT " "
END SELECT

PRINT " "
INPUT "press any key to continue$";carryOn$
cls
END SUB



'***improvement: use a character$ to tell program if rep or
  not, combine SUBS into one
SUB ReplicateBoundsInfringements ( repBoundsType%,
  repxBoundByMin&, repxBoundByMax&, repxBoundByHi95CI&,
  repxBoundByLo95CI&, minRepxIsUnder95CI&,
  maxRepxIsOver95CI&,_

  repyBoundByMin&, repyBoundByMax&, repyBoundByHi95CI&,
  repyBoundByLo95CI&, minRepyIsUnder95CI&,
  maxRepyIsOver95CI&)
REM prints the number of times that bounds for simulated
  meanX or meanY are crossed.

SHARED numPoints%, sampleSizeX%(), sampleSizeY%(),
  numTrials%, simulationsForX&, simulationsForY&,
  loopCounter&
LOCAL carryOn$

Select Case repBoundsType%

CASE 0
 PRINT "no bounds set on simulation of replicates."

CASE 1
 PRINT " "
 PRINT "TALLY: ";simulationsForX&;" simulations of rep x /
   ";simulationsForY&;" simulations of rep y"
 PRINT
"-----------------------------------------------------------
----------------"
 PRINT " "
 PRINT "initial simulated replicate x is bound by min: (";
```

```
    repxBoundByMin&;") times."
  PRINT "intiial simulated repx is bound by (Hi/Lo) 95% CI:
    ("; repxBoundByHi95CI& ;"/"; repxBoundByLo95CI&;")
      times."
  PRINT "arbitrary min repx is under 95% CI:
    (";minRepxIsUnder95CI&;") times."
  PRINT " "

  PRINT "initial simulated replicate y is bound by min: (";
    repyBoundByMin&;") times. "
  PRINT "initial simulated repy is bound by (Hi/Lo) 95% CI:
    ("; repyBoundByHi95CI&;"/"; repyBoundByLo95CI&;") times."
  PRINT "arbitrary min repy is under 95% CI:
    (";minRepyIsUnder95CI&;") times."
  PRINT " "

CASE 2,3
  PRINT " "
  PRINT "TALLY: ";simulationsForX&;" simulations of rep x /
    ";simulationsForY&;" simulations of rep y"
  PRINT
"-----------------------------------------------------------
-----------------"
  PRINT " "
  PRINT "initial simulated replicate x is bound by (max/min):
    ("; repxBoundByMax&;"/";repxBoundByMin&;") times."
  PRINT "intiial simulated repx is bound by (Hi/Lo) 95% CI:
    ("; repxBoundByHi95CI& ;"/"; repxBoundByLo95CI&;")
      times."
  PRINT "max repx and min repx are (over/under) 95% CI:
    (";maxRepxIsOver95CI&;"/";minRepxIsUnder95CI&;") times."
  PRINT " "

  PRINT "initial simulated repy is bound by (max/min): (";
    repyBoundByMax&;"/";repyBoundByMin&;") times. "
  PRINT "initial simulated repy is bound by (Hi/Lo) 95% CI:
    ("; repyBoundByHi95CI&;"/"; repyBoundByLo95CI&;") times."
  PRINT "max repy and min repy are (over/under) 95% CI:
    (";maxRepyIsOver95CI&;"/";minRepyIsUnder95CI&;") times."
  PRINT " "
END SELECT

PRINT " "
INPUT "press any key to continue"; carryOn$
cls
END SUB
```

Computer code for the logNormal replicate program

The logNormal replicate program closely resembled the CLT program, and only that code which differs from the CLT program is included here. SUB FindConfidenceIntervals and SUB Randomization have counterparts in the CLT program, but SUB MakeDistOfMeans is not in the CLT program. There is also an additional parameter that the logNormal replicate program requires: the number of means that must be generated to estimate confidence intervals for means. This is one of the first values that a user is asked to give by keyboard, and is stored in "overPop%" (the code for this is not shown).

```
SUB FindConfidenceIntervals (W(1), numPoints%,
  sampleSizeW%(1), S2W(1), lowerCI(1), upperCI(1))
REM finds confidence intervals for each mean W using
  mean:variance or real data S2.
LOCAL S2, i
SHARED overPop%, zmean()    ' desired number of means taken
                              from under pop.

 FOR i = 1 to numPoints%     ' for each lake
   dmean = W(i)       'desired mean of logNormal dist.
                       (ie. sample mean.)
   dvar  = S2W(i)     'desired variance of logNormal dist.
                       (ie. sample var.)
   k  = LOG( dmean )
   rNstd = SQR( LOG( dvar+EXP( 2*k ) ) - (2*K) )
     'Normal Std required to produce desired logN dist.
   rNmean = LOG( dMean ) - ((rNstd^2)/2)
     'Normal mean required to produce desired logN dist.

   CALL MakeDistOfMeans (overPop%, sampleSizeW%(i), rNmean,
     rNstd, zmean())

   low  = INT ( .025 * overPop% )      'lower tail of 95% CI
   high = INT (.975 * overPop% )
   if low < 1 then low = 1
   upperCI(i) = zmean(high)
   lowerCI(i) = zmean(low)
 NEXT i
END SUB


SUB MakeDistOfMeans (overPop%, sampleSize%, rNmean, rNstd,
  meanz(1))
REM samples overPop% # of means from a logNormal dist.and
  puts them
REM in an array  called meanz(), sorted by QuickSort.
```

```
LOCAL  i, sum, j, x, z, low, high

  FOR i = 1 to overPop%
     sum = 0
     FOR j = 1 to sampleSize%
        rand = RND
        x = (.5513 * LOG( rand/(1-rand)) ) * rNstd + rNmean
           'x is Ncrm(rNmean,rNstd^2)
        z = EXP (x)    'z is Lnorm(dMean, dvar)
        sum = sum + z
        NEXT j
     meanz(i) = sum/sampleSize%
     NEXT i
  CALL QuickSort ( 1, overPop%, meanz() )
END SUB


SUB Randomization (dvar, sampleSize%, dMean, simulatedValue)
REM  Randomly samples a value from a Normal (meanValue,
  SQR(S2/n)) dist.
REM  Note that it is Non-transformed values of x and y that
  are manipulated.
LOCAL SE, rand, varMean

 k = LOG( dMean )
 rNstd = SQR( LOG( dvar+EXP( 2*k ) ) - (2*K) ) 'Normal Std
   required to produce desired logN dist.
 rNmean = LOG( dMean ) - ((rNstd^2)/2)          ' Normal mean

  sum = 0
  FOR j = 1 to sampleSize%       'sample from under dist.
    rand = RND
    simNorm  = (.5513 * LOG( rand / (1-rand) )) * rNstd +
      rNmean  'normal dist.
    simLogNorm  = EXP (simNorm)       'logNorm
    sum = sum + simLogNorm
  NEXT
  simulatedValue = sum/sampleSize%
END SUB
```

Computer code for the Normal replicate program

The Normal replicate program and the logNormal replicate program were the same, except in the way that they generated distributions of replicates. The Normal replicate program used an equation that generated values from a Normal distribution. These differences appear in the three procedures shown below, whose counterparts in the logNormal replicate program are shown above.

```
SUB FindConfidenceIntervals (W(1), numPoints%,
  sampleSizeW%(1), S2W(1), lowerCI(1), upperCI(1))
REM finds confidence intervals for each mean W using
  mean:variance or real data S2.
LOCAL S2, i
SHARED overPop%, zmean()    ' desired number of means taken
                              from under pop.

 FOR i = 1 to numPoints%     ' for each lake
   std = SQR ( S2W(i) )
   CALL MakeDistOfMeans (overPop%, sampleSizeW%(i), W(i),
     std, zmean())

   low  = INT ( .025 * overPop% )      'lower tail of 95% CI
   high = INT (.975 * overPop% )
   if low < 1 then low = 1
   upperCI(i) = zmean(high)
   lowerCI(i) = zmean(low)
 NEXT i
END SUB


SUB MakeDistOfMeans (overPop%, sampleSize%, mean, std,
  meanx(1))
REM samples overPop% # of means from a logNormal dist.and
  puts them
REM in an array  called meanx(), sorted by QuickSort.

LOCAL  i, sum, j, x, low, high

  FOR i = 1 to overPop%
     sum = 0
     FOR j = 1 to sampleSize%
        rand = RND
        x = (.5513 * LOG( rand/(1-rand)) ) * std + mean
          'x is Norm(mean,std^2)
        sum = sum + x
        NEXT j
     meanx(i) = sum/sampleSize%
```

```
    NEXT i
  CALL QuickSort ( 1, overPop%, meanx() )
END SUB


SUB Randomization (var, sampleSize%, mean, simulatedValue)
REM  Randomly samples a value from a Normal (meanValue,
  SQR(S2/n)) dist.
REM  Note that it is Non-transformed values of x and y that
  are manipulated.
LOCAL std, rand, varMean

  std = SQR ( var )
  sum = 0
  FOR j = 1 to sampleSize%       'sample from under dist.
    rand = RND
    simNorm  = (.5513 * LOG( rand / (1-rand) )) * std + mean
      'normal dist.
    sum = sum + simNorm
  NEXT
  simulatedValue = sum/sampleSize%
END SUB
```