Simulations for Research Design

William Trochim

Cornell University

Sarita Davis

Georgia State University

This is a complete workbook that introduces the use of computer simulations in applied social research designs. There are two versions of each of the simulations, one accomplished manually (by rolling dice) and the other done using the MINITAB statistical package. The major two-group, program-comparison designs (randomized experiment, regression-discontinuity, nonequivalent group design) are simulated and the issue of regression artifacts is studied. Please direct any comments to Bill Trochim.

The online version of this workbook can be found at
http://www.socialresearchmethods.net/simul/simul.htm

Contents

Acknowledgments

These simulation exercises have evolved from an earlier set of dice rolling exercises that Donald T. Campbell used (and still uses, we hear) in the 1970s in teaching research methodology to undergraduate and graduate students. Over the years, those exercises helped introduce many a struggling graduate student to the joys of both simulation and methodology. We hope that much of the spirit of those earlier simulations is retained here. Certainly, none of the problems in our simulations can be attributed to Campbell's efforts. He was able to achieve a blend of congeniality and rigor that we have tried to emulate.

The computer versions of these simulations came out of Bill Trochim's efforts in the early 1980s to translate some of those Campbell dice rolling exercises into increasingly available computer technologies. Previous versions were implemented in a number of the graduate and undergraduate research methods courses at Cornell over the years. We owe a great debt to the many students who struggled with earlier drafts and offered their valuable criticisms and suggestions.

During the mid-80s Trochim began working with these exercises with James Davis who, at the time, was T.A. for his graduate-level methods courses. James improved on them considerably, taking what were separate exercises and integrating them into a single computerized simulation that illustrated the three major pre/post research designs. His efforts led to two co-authored articles on simulation cited in this workbook.

This current set of exercises was resurrected in the Spring of 1993 initially to provide an interesting and challenging problem area for Sarita Tyler's Ph.D. qualifying examinations. Essentially she took a set of file folders that had some poorly xeroxed copies of the old dice rolling and computer exercises on them, and integrated these into the coherant package contained here. We had no idea when she began that this process was going to result in an integrated workbook -- all she originally intended was to learn something about simulations. Clearly the present volume would not have happened without her considerable efforts.

Introduction to Simulations

Simulation (sim' yoo la 'shen) an imitation or counterfeit. This definition, according to Websters Dictionary, implies the presence of a replication so well constructed that the product can pass for the real thing. When applied to the study of research design, simulations can serve as a suitable substitute for constructing and understanding field research. Trochim and Davis (1986) posit that simulations are useful for (1) improving student understanding of basic research principles and analytic techniques: (2) investigating the effects of problems that arise in the implementation of research; and (3) exploring the accuracy and utility of novel analytic techniques applied to problematic data structures.

As applied to the study of research design, simulations can serve as a tool to help the teacher, evaluator, and methodologist address the complex interaction of data construction and analysis, statistical theory, and the violation of key assumptions. In a simulation, the analyst first creates data according to a known model and then examines how well the model can be detected through data analysis. Teachers can show students that measurement, sampling, design, and analysis issues are dependent on the model that is assessed. Students can directly manipulate the simulation model and try things out to see immediately how results change and how analyses are affected. The evaluator can construct models of evaluation problems -- making assumptions about the pretest or type of attrition, group nonequivalence, or program implementation -- and see whether the results of any data analyses are seriously distorted. The methodologist can systematically violate assumptions of statistical procedures and immediately assess the degree to which the estimates of program effect are biased (Trochim and Davis, 1986, p. 611).

Simulations are better for some purposes than is the analysis of real data. With real data, the analyst never perfectly knows the real-world processes that caused the particular measured values to occur. In a simulation, the analyst controls all of the factors making up the data and can manipulate these systematically to see directly how specific problems and assumptions affect the analysis. Simulations also have some advantages over abstract theorizing about research issues. They enable the analyst to come into direct contact with the assumptions that are made and to develop a concrete "feel" for their implications on different techniques.

Simulations have been widely used in contemporary social research (Guetzkow, 1962; Bradley, 1977, Heckman, 1981). They have been used in program evaluation contexts, but to a much lesser degree (Mandeville, 1978; Raffeld et at., 1979; Mandell and Blair 1980). Most of this work has been confined to the more technical literature in these fields.

Although the simulations described here can certainly be accomplished on mainframe computers, this workbook will illustrate their use in manual and microcomputer contexts. There are several advantages to using simulations in these two contexts. The major advantage to manual simulations is that they cost almost nothing to implement. The materials needed for this process are: dice, paper, and pencils. Computer simulations are also relatively low in cost. Once you have purchased the microcomputer and necessary software there are virtually no additional costs for running as many simulations as are desired. As it is often advantageous to have a large number of runs of any simulation problem, the costs in mainframe computer time can become prohibitive. A second advantage is the portability and accessibility. Manual simulations can be conducted anywhere there is a flat surface on which to roll dice. Microcomputers are also portable in that one can easily move from home to office to classroom or into

an agency either to conduct the simulations or to illustrate their use. Students increasingly arrive at colleges and universities with microcomputers that enable them to conduct simulations on their own.

This workbook illustrates some basic principles of manual and computer simulations and shows how they may be used to improve the work of teachers, evaluators, and methodologists. The series of exercises contained in this manual are designed to illuminate a number of concepts that are important in contemporary social research methodology including:

- simulations and their role in research
- basic measurement theory concepts
- the elements of pretest/posttest group designs, including nonequivalent, regression-discontinuity and randomized experimental designs
- some major threats to internal validity, especially regression artifacts and selection threats

The basic model for research design presented in this simulation workbook is the program or outcome evaluation. In program evaluation the goal is to assess the effect or impact of some program on the participants. Typically, two groups are studied. One group (the program group) receives the program while the other does not (the comparison group). Measurements of both groups are gathered before and after the program. The effect of the program is determined by looking at whether the program group gains more than the comparison group from pretest to posttest. The exercises in this workbook describe how to simulate the three most commonly used program evaluation designs, the Randomized Experiment, the pretest/posttest Nonequivalent Group Design, and the Regression-Discontinuity design. Additional exercises are presented on regression artifacts, which can pose serious threats to internal validity in research designs that involve within-subject treatment comparisons.

We can differentiate between these research designs by considering the way in which assignment of units to treatment conditions is conducted - in other words, what rule has determined treatment assignment. In the randomized experimental (RE) design, persons are randomly assigned to either the program or comparison group. In the regression-discontinuity (RD) design (Trochim, 1984), all persons who score on one side of a chosen preprogram measure cutoff value are assigned to one group, with the remaining persons being assigned to the other. In the nonequivalent group design (NEGD) (Cook and Campbell, 1979; Reichardt, 1979), persons or intact groups (classes, wards, jails) are "arbitrarily" assigned to either the program or comparison condition. These designs have been used extensively in program evaluations where one is interested in determining whether the program had an effect on one or more outcome measures. The technical literature on these designs is extensive (see for instance, Cook and Campbell, 1979; Trochim, 1986). The general wisdom is that if one is interested in establishing a causal relationship (for example, in internal validity), RE designs are most preferred, the RD design (because of its clear assignment-by-cutoff rule) is next in order of preference, and the NEGD is least preferable.

All three of the program evaluation designs (RE, RD, and NEGD) have a similar structure, which can be described using the notation:

O X O

O O

where the Os indicate measures and the X indicates that a program is administered. Each line represents a different group; the first line depicts the program participants whereas the second shows the comparison group. The passage of time is indicated by movement from left to right on a line. Thus, the program group is given a preprogram measure (indicated by the first O), is then given the program (X), and afterward is given the postprogram measure (the last O). The vertical similarity in the measurement structure implies that both the pre and postmeasures are given to both groups at the same time. Model-building considerations will be discussed separately for each design.

The simulations are presented in two parts. The first part contains the manual simulations, including the basic randomized experiment, nonequivalent group and regression-discontinuity research designs with an additional exercise presented on regression artifacts. Part two of this manual contains the computer simulation equivalents of the research designs presented in part one. Also included in this section is a computer analog to the regression artifacts simulation.

Both Parts I and II begin with an exercise called Generating Data. This exercise describes how to construct the data that will be used in subsequent exercises. Because this exercise lays the foundation on which subsequent simulations are based, it is extremely important that you do it first and follow the instructions very carefully.

Manual Simulations

Generating Data

This exercise will illustrate how simulated data can be created by rolling dice to generate random numbers. The data you create in this exercise will be used in all of the subsequent manual simulation exercises. Think about some test or measure that you might like to take on a group of individuals. You administer the test and observe a single numerical score for each person. This score might be the number of questions the person answered correctly or the average of their ratings on a set of attitude items, or something like that, depending on what you are trying to measure. However you measure it, each individual has a single number that represents their performance on that measure. In a simulation, the idea is that you want to create, for a number of imaginary people, hypothetical test scores that look like the kinds of scores you might obtain if you actually measured these people. To do this, you will generate data according to a simple measurement model, called the "true score" model. This model assumes that any observed score, such as a pretest or a posttest score, is made up of two components: true ability and random error. You don't see these two components when you measure people in real life, you just assume that they are there.

We can describe the measurement model with the formula

$$O = T + e_o$$

where O is the observed score, T is the person's true ability or response level on the characteristic being measured and e_o represents random error on this measure. In real life, all we see is the person's score -- the O in our formula above. We assume that part of this number or score tells us about the true ability or attitude of the person on that measure. But, we also assume that part of what we observe in their score may reflect things other than what we are trying to measure. We call this the error in measurement and use the symbol e_o to represent it in the formula. This error reflects all the situational factors (e.g., bad lighting, not enough sleep the night before, noise in the testing room, lucky guesses, etc.) which can cause a person to score higher or lower on the test than his/her true ability or level alone would yield. In the true score measurement model, we assume that this error is random in nature, that for any individual these factors are as likely to inflate or deflate their observed score. There are models for simulating data that make different assumptions about what influences observed scores, but the true score model is one of the simplest and is the most commonly assumed.

You will use this true score model to generate imaginary pretest and posttest scores for 50 hypothetical persons. This will be accomplished using a pair of dice. For each person you will roll the pair of dice once to generate a score representing true ability, once to generate pretest measurement error and once to generate posttest measurement error. These values should be entered for each person in the appropriate columns in Table 1-1. You will then construct a pretest using the simple formula

$$X = T + e_X$$

where X is the pretest, T is the true ability (simply the sum of the roll of a pair of dice) and e_X is pretest measurement error (also based on the sum of the roll of a pair of dice). In real life this is all you would be given, and you would assume that each test score is a reflection of some true ability and random error. You would not see the two components; you only see the observed score. Similarly, you will then construct a posttest score using the formula

$$Y = T + e_Y$$

where Y is the posttest, T the same true score that is used for the pretest and $e_Y$is posttest measurement error (based on the sum of yet another roll of the pair of dice).

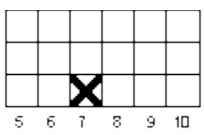
This procedure can be made clearer by doing it. Notice that the first column in Table 1-1 lists the numbers of the persons in the study, from 1 to 50. You will begin by generating a pretest and posttest score for person 1. First, roll the pair of dice once and sum the values (this will be a score between 2 and 12). This is called the true score. Enter the value in the first row of column 2. This score represents the true ability or level (T) of person 1 on this measure. Repeat this step for all 50 persons.

Second, roll two dice and place their sum in the first row of column 3. This number represents the error in measurement on the pretest (e_X). Repeat this for all 50 persons. Third, roll the pair of dice again and place their sum in the first row of column 4. This value represents error in measurement on the posttest (e_Y). Again, repeat this for all 50 persons. You have now created an imaginary true score and errors in measurement for all 50 persons, recording the results in the appropriate columns.

Now you are going to construct the observed pretest and posttest scores. This requires only simple addition. For the first person (row) take the true score (T) from column 2 and add it to the pretest error value (e_X) from column 3. Place this sum in column 5 (the pretest, X). Do this for all 50 people. Now, for the first person, add the true score (T) from column 2 to the posttest error value (e_Y) from column 4. Place this sum in column 6 (the posttest, Y). Do this for all 50 people.

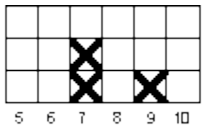
It would be worth stopping at this point to think about what you have done. You have been creating imaginary test scores. You have constructed two tests called X and Y. Both of these imaginary tests measure the same trait because both of them share the same true score. The true score reflects the true ability of each person on this imaginary or simulated test. In addition, each test has its own random error. If this were real life, of course, you would not be constructing test scores like this. Instead, you would simply be given the two sets of observed test scores, X and Y. You would assume that the two measures have a common true score and independent errors but would not see these. Thus, you have generated simulated data. The advantage of using such data is that, unlike with real data, you know how the X and Y tests are constructed because you constructed them. You will see in later simulations that this enables you to test different analyses to see if they give the results that you put into the data. If the analyses work on simulated data, then, you may assume that they will also work for real data as long as the real data meet the assumptions of the measurement model used in the simulations.

Next, you are going to look at the pretest and posttest data you simulated. Let's do this by graphing the pretest and posttest histograms. Figure 1-1 can be used to graph the pretest. Begin with the first person's pretest (X) value in column 5. Locate the column on Figure 1-1 for that value and make an 'X' in the first row of that column on the figure. For instance, if the first person has a pretest score of 7, your graph should look like:



Now continue plotting the pretest values for the 50 people. If you come to a value that you already had before, place your 'X' in the row above the last 'X' you made for that value. For instance, if the second

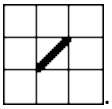
person had a pretest score of 9 and the third had a score of 7, your graph for these first three people would look like:

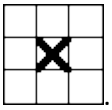


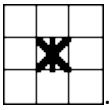
Repeat this for the pretest scores for all 50 people. Now, using Figure 1-2, repeat this process to draw the histogram for the posttest values in column 6.

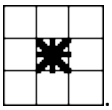
Now let's estimate the central tendency for the pretest distribution shown in Figure 1-1. The best way to do this would be to calculate the mean or average of the 50 scores. But a quicker way to get a rough idea would be to locate the middle of the distribution by counting. Starting with the lowest column in which there is an 'X' in Figure 1-1, count the lowest 25 'Xs' in the figure. What column of Figure 1-1 is the 25th 'X' in? Simply put a mark at the bottom of the figure under this column to show that this is where the "center" of the distribution is located. Then, use the same counting procedure to estimate where the center is on the posttest histogram of Figure 1-2.

Now, let's look at the pretest and posttest scores together. You will graph their bivariate (i.e., two-variable) distribution on the graph in Figure 1-3. To do this, begin with the pretest and posttest score for person 1. Notice that the pretest is shown on the horizontal axis while the posttest is the vertical one. Go along the horizontal axis in Figure 1-3 until you come to the value for the pretest score for the first person. Now go up in that column until you come to the row that has the value for the posttest score for the first person. You are going to make a mark in the box that represents the pretest (column) and posttest (row) value for the first person. But because there may be more than one person who has the same pretest and posttest score, you will want to use a system to mark the box that allows you to see how many people of the fifty have a pre-post pair in any box. We recommend that you use the following system.

For the <u>first</u> mark in a specific box, do  .

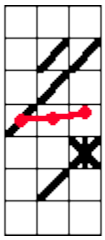
The second time you find a person with the same pre/post pair, add another diagonal  .

For a third case, add a vertical line  .

If there is a fourth, add a horizontal line  .

It is not likely that you will have any more than four cases in any given box, but if you do, create a way to indicate this. In this manner, plot all of the pre/post pairs for the 50 persons in your simulation.

Now let's try to fit a line through this bivariate distribution in Figure 1-3. To do this, begin with the leftmost column on the graph. For each column, you are going to try to estimate its central tendency. If there are no marks in a column, skip that column and move to the next column to the right. If there are marks in the column, place a dot (●) halfway between the lowest and highest mark in that column. If there is only one mark in a column, just place the dot in that row. Note that there will only be one dot per column. (This is, admittedly, a rough and simplified way to estimate central tendency. If you want to be more accurate, you can calculate the average posttest score for all persons having the same pretest score and place your mark accordingly.) Nevertheless, our rough estimation procedure should approximate the central tendency well enough for our purposes here. Now, beginning with the dot farthest to the left, connect the dots in adjacent columns with a line. Because it may be hard to distinguish this line from the bivariate marks you made in the boxes, you might want to connect these dots using a different colored pen. The figure below shows how a part of your bivariate plot with the dots and connecting lines might look..



Is the line that connects the dots in your graph relatively smooth? or very jagged? Is it a flat (horizontal) line? or not? Does this line tell you anything about the relationship between the pretest and posttest? It should be clear that the X and Y tests are positively related to each other, that is, higher scores on one test tend to be associated with higher scores on the other.

Now, you should again stop to consider what you have done. In the first part of the exercise you generated two imaginary tests--X and Y. In the second part, the bivariate graph showed you that the two tests are positively related to each other. You set them up to be related by including the same true ability score in both tests. You should think about the following points:

- If you had generated data for thousands of persons, the pretest and posttest distributions would look nearly identical. Furthermore, the estimates of pretest and posttest central tendency (e.g., averages) would be nearly identical and both distributions would have equal numbers of persons on either side of the central values. You can get a better sense of this if you compare your graphs with those of other persons who do this exercise.
- Each score (pretest and posttest) is composed of equal parts of true ability and random error. This is a common (although simplistic) measurement model called the "true score" model. Because we only have one true score for each test, we are assuming that each test is unidimensional, that is, measures only one trait. A factor analysis of both tests should yield one factor.
- The amounts of true score and error which are present in a test determine the reliability of the test. If you had used two parts true score to one part error, you would have more reliable tests; if you had used one part true score to two parts error, less reliable tests. (Specifically, reliability is defined as the ratio of the variance of the true scores to the variance of the total or observed score.)

- The pretest and posttest are related because they both share the same true score. (If you had generated separate pretest and posttest true scores there would be no relationship between the two tests.) But the relationship between pretest and posttest is far from perfect because each test has independent measurement error. In this example, if you computed the correlation it would be about .5.
- The line that you fit to the bivariate distribution is a very rough approximation to a regression line. You should be convinced that if you had thousands of persons, the best line through the data would be a straight line with a slope equal to about .5. (If the variances of the two variables are equal, as in this example, the correlation would be equal to the slope of the regression line. You can see whether the variances appear equal by looking at the spread of the scores around the central values in the pretest and posttest frequency distributions.)

Generating Data

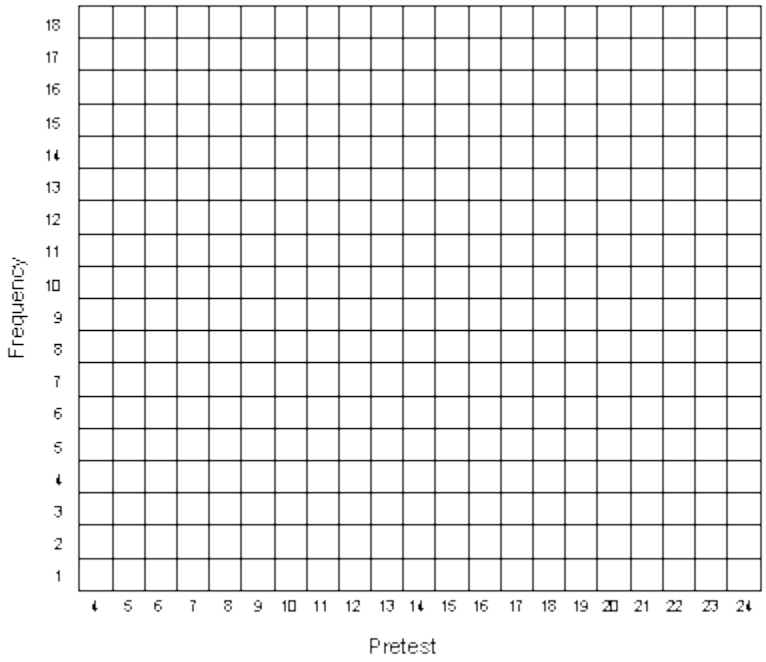
Table 1-1

1	2	3	4	5	6
Person	True Score (T)	Pretest Error (e_X)	Posttest Error (e_Y)	Pretest X	Posttest Y
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

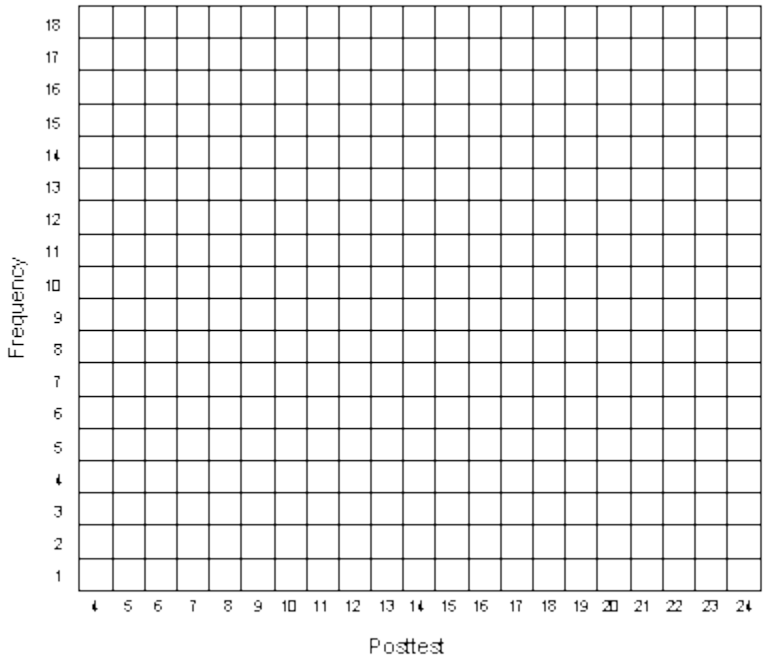
Table 1-1 (cont.)

1	2	3	4	5	6
Person	True Score (T)	Pretest Error (e_X)	Posttest Error (e_Y)	Pretest X	Posttest Y
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					
46					
47					
48					
49					
50					

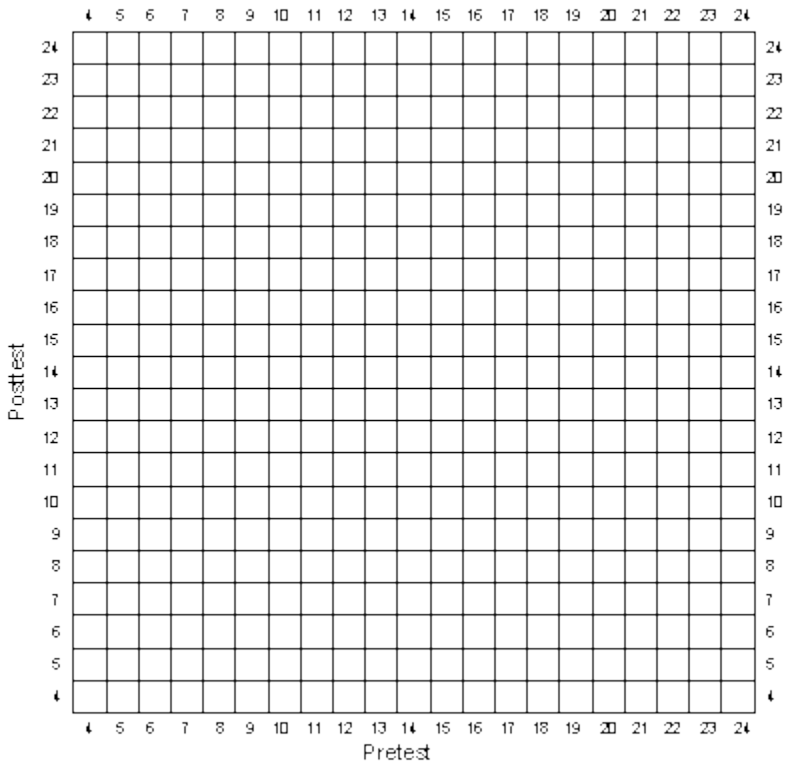
Generating Data
Figure 1-1



Generating Data
Figure 1-2



Generating Data
Figure 1-3



The Randomized Experimental Design

In this exercise you will simulate a simple pre/post randomized experimental design. This design can be depicted in notational form as

R O X O
R O O

where each O indicates an observation or measure on a group of people, the X indicates the implementation of some treatment or program, separate lines are used to depict the two groups in the study, the R indicates that persons were randomly assigned to either the treatment or control group, and the passage of time is indicated by moving from left to right. We will assume that we are comparing a program and comparison group (instead of two programs or different levels of the same program).

You will use the data you created in the first exercise. Copy the pretest scores from the first exercise (Table 1-1, column 5) into column 2 of Table 2-1. Now we need to randomly assign the 50 persons into two groups. To do this, roll one die for each person. If you get a 1,2, or 3, consider that person to be in the program group and place a Ô1Õ in the column 3 of Table 2-1 labeled ÒGroup Assignment (Z)Ó. If you get a 4, 5, or 6, consider that person to be in the comparison group and place a ‘0’ in the column 3. Now, imagine that you give the program or treatment to the people who have a '1' for Group Assignment and that the program has a positive effect. In this simulation, we will assume that the program has an effect of 7 points for each person who receives it. This "Hypothetical Program Effect" is shown in column 4 of Table 2-1. To determine the treatment effect for each person, multiply column 3 by column 4 and place the result in column 5 labeled ÒEffect of Program (G)Ó. (The G stands for how much each person Gains as a result of the program.) You should have a value of Ô7Õ for all persons randomly assigned to the program group and a value of Ô0Õ for those in the comparison group. Why did we multiply the column of 0 and 1 values by the column of 7s when we could just as easily have told you to simply put a 7 next to each program recipient? We do it this way because it illustrates how a 0,1 variable, called a "dummy variable," can be used in a simple formula to show how the Effect of the Program is created. In this simulation, for instance, we can summarize how the program effect is created using the formula

$$G = Z \times 7$$

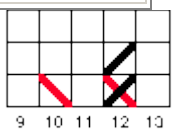
where G is the gain or program effect, Z is the dummy coded (0,1) variable in column 3 and the 7 represents the constant shown in column 5.

Next, copy the posttest values (Y) from the first exercise (Table 1-1, column 6) to column 6 of Table 2-1 labeled “Posttest (Y) from Table 1-1". Finally, to create the observed posttest value that has a program effect built in, add the values in column 5 and 6 in Table 2-1 and put them into column 7 labeled "Posttest (Y) for Randomized Experimental DesignÓ. You should recognize that this last column in Table 2-1 has the same posttest scores as in the first simulation exercise, except that each randomly assigned program person has 7 extra points that represent the effect or gain of the program.

As before, you should graph the univariate distributions for the pretest and posttest in Figures 2-1 and 2-2. But here, unlike in the first exercise, the 50 people are randomly divided into two groups. It would be nice if you could distinguish the scores of these two groups in your graphs of the distributions. You should do this by using different colored pens or pencils and lines that slant in different directions for

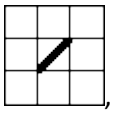
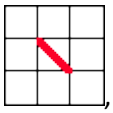
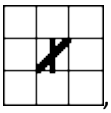
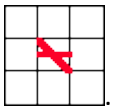
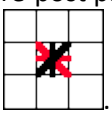
the program and comparison cases. For instance, let's say that the first four persons in Table 2-1 have the following pretest scores and group assignments in Table 2-1:

Person	Pretest X from Table 1-1	Group Assignment Z
1	12	1
2	12	0
3	10	0
4	12	1

In this case, the histogram for these four persons would look like 

Now plot the data for all 50 persons for both the pretest (Figure 2-1) and posttest (Figure 2-2) making sure to distinguish between the two randomly assigned groups both color and in the angle of the mark you make. As in the first simulation, you should estimate the central tendency for both the pretest and posttest, but here you should do it separately for the program and comparison cases.

Now, plot the bivariate distribution in Figure 2-3. Again, you need to have a system for distinguishing between program and comparison cases when they fall in the same box. Use different colored pens or pencils for each. In addition, use different marks according to the following system:

First program case  , first comparison case  , second program case  , second comparison case  . So, if you have a pre-post pair that happens to have two program and two comparison cases, the graph should look like  . Now, plot the lines on the bivariate plot that describe the pre/post relationship (as described in the first simulation exercise), doing a separate line for the program and comparison groups.

There are several things you should note. First, the pretest central values for the two groups should be similar (although they are not likely to be exactly the same). This is, of course, because the groups were randomly assigned. Second, the posttest central values should clearly differ. In fact, we expect that the difference between the average posttest values should be approximately the 7 units that you put in. In addition, the vertical difference between the relationship lines in the bivariate plot (Figure 2-3) should also be about 7 units.

At this point you should be convinced of the following:

- The design simulated here is a very simple single-factor randomized experiment. The single factor is represented by the 0,1 treatment variable that represents which group people are in. You could simulate more complex designs. For example, to simulate a randomized block design you would first rank all persons on the pretest. Then you could set the block size, for example at n = 2. Then, beginning with the lowest two pretest scorers, you would randomly assign one to the program group and the other to the comparison group. You could do this by rolling a die--if you get a 1, 2, or 3 the lowest scorer is a program participant; if you get a 4, 5, or 6 the higher scorer is. Continuing in this manner for all twenty-five pairs would result in a block design.
- You could also simulate a 2 x 2 factorial design. Let's say that you wanted to evaluate an educational program that was implemented in several different ways. You decide to manipulate the manner in which you teach the material (Lecture versus Printed Material) and where the program is given (In-class or Pull-out). Here you have two factors -- Delivery of Material and Location -- and each factor has two levels. We could summarize this design with a simple table that shows the four different program variations you are evaluating

		Method Of Delivery	
		Lecture	Printed Materials
Location	In-Class	Program 1	Program 2
	Pull-Out	Program 3	Program 4

Notice that we need a 2x2 table to summarize this study. Not surprisingly, we would call this a 2x2 factorial experimental design. If you were simulating this, you would actually have to randomly assign persons into one of the four groups. One advantage of this kind of design is that it allows you to examine how different program characteristics operate together (or "interact") to produce a program effect.

- You should recognize that the design simulated here can be considered a repeated measures experimental design because a pretest and posttest are used. You could simulate a posttest-only experimental design as well.

Randomized Experimental Design
Table 2-1

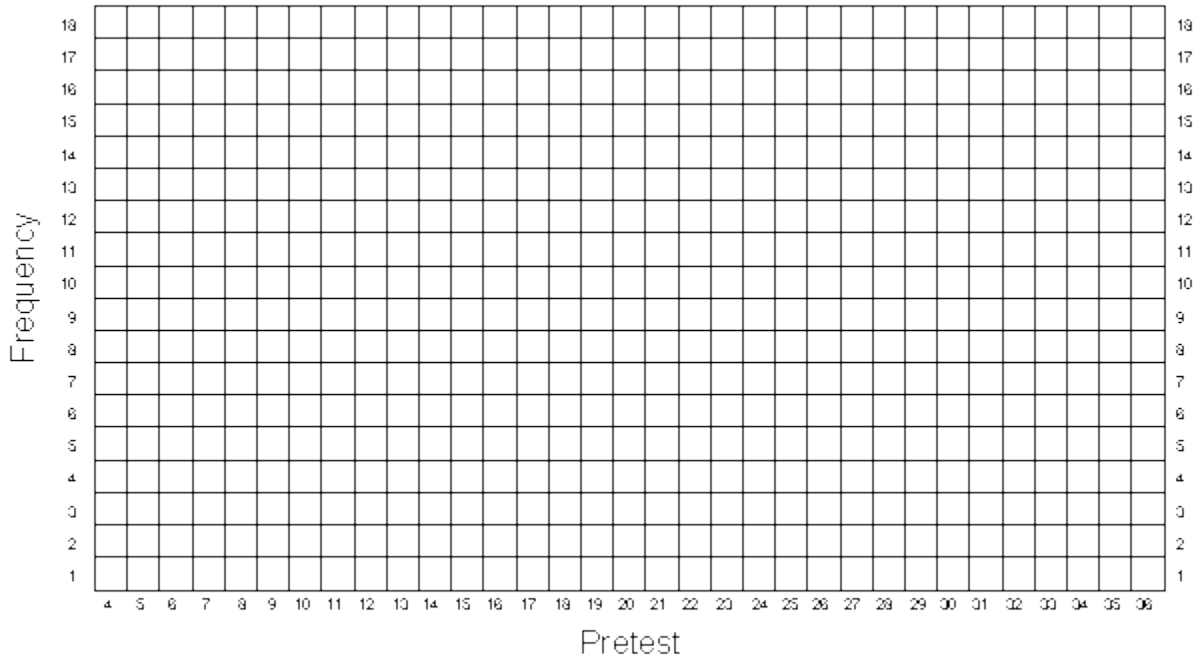
1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Group Assignment Z	Hypothetical Program Effect	Effect of Program (G)	Posttest Y from Table 1-1	Posttest (Y) for Randomized Experimental Design
1			7			
2			7			
3			7			
4			7			
5			7			
6			7			
7			7			
8			7			
9			7			
10			7			
11			7			
12			7			
13			7			
14			7			
15			7			
16			7			
17			7			
18			7			
19			7			
20			7			
21			7			
22			7			
23			7			
24			7			
25			7			

Randomized Experimental Design
Table 2-1 (cont.)

1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Group Assignment Z	Hypothetical Program Effect	Effect of Program (G)	Posttest Y from Table 1-1	Posttest (Y) for Randomized Experimental Design
26			7			
27			7			
28			7			
29			7			
30			7			
31			7			
32			7			
33			7			
34			7			
35			7			
36			7			
37			7			
38			7			
39			7			
40			7			
41			7			
42			7			
43			7			
44			7			
45			7			
46			7			
47			7			
48			7			
49			7			
50			7			

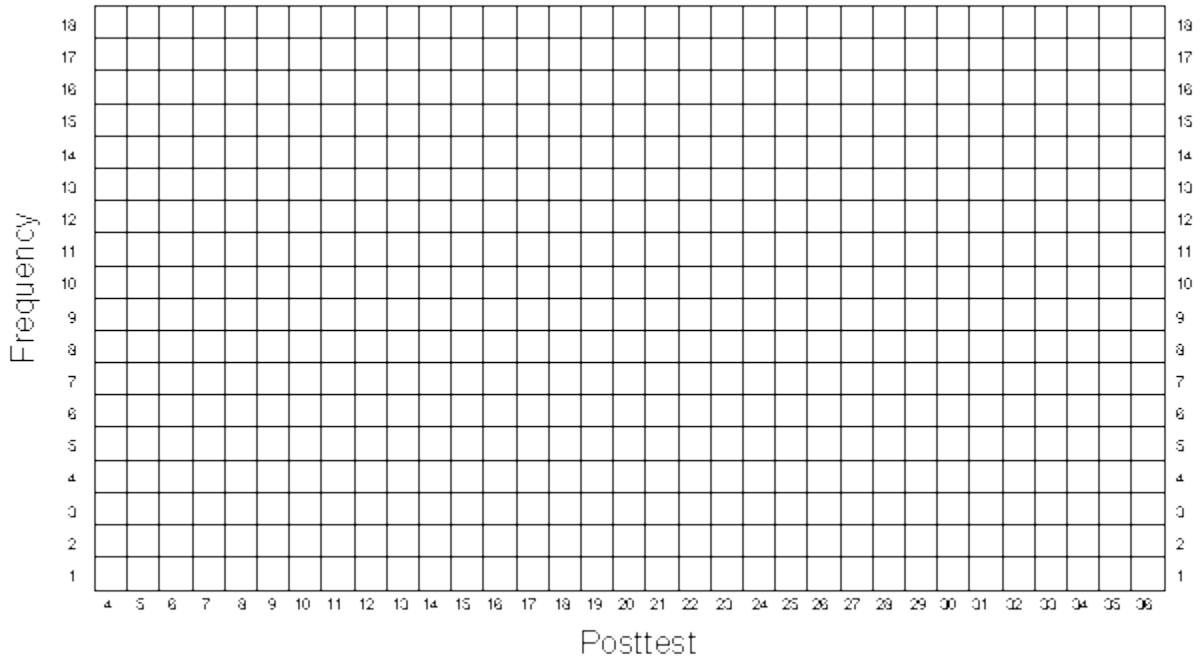
Randomized Experimental Design

Figure 2-1



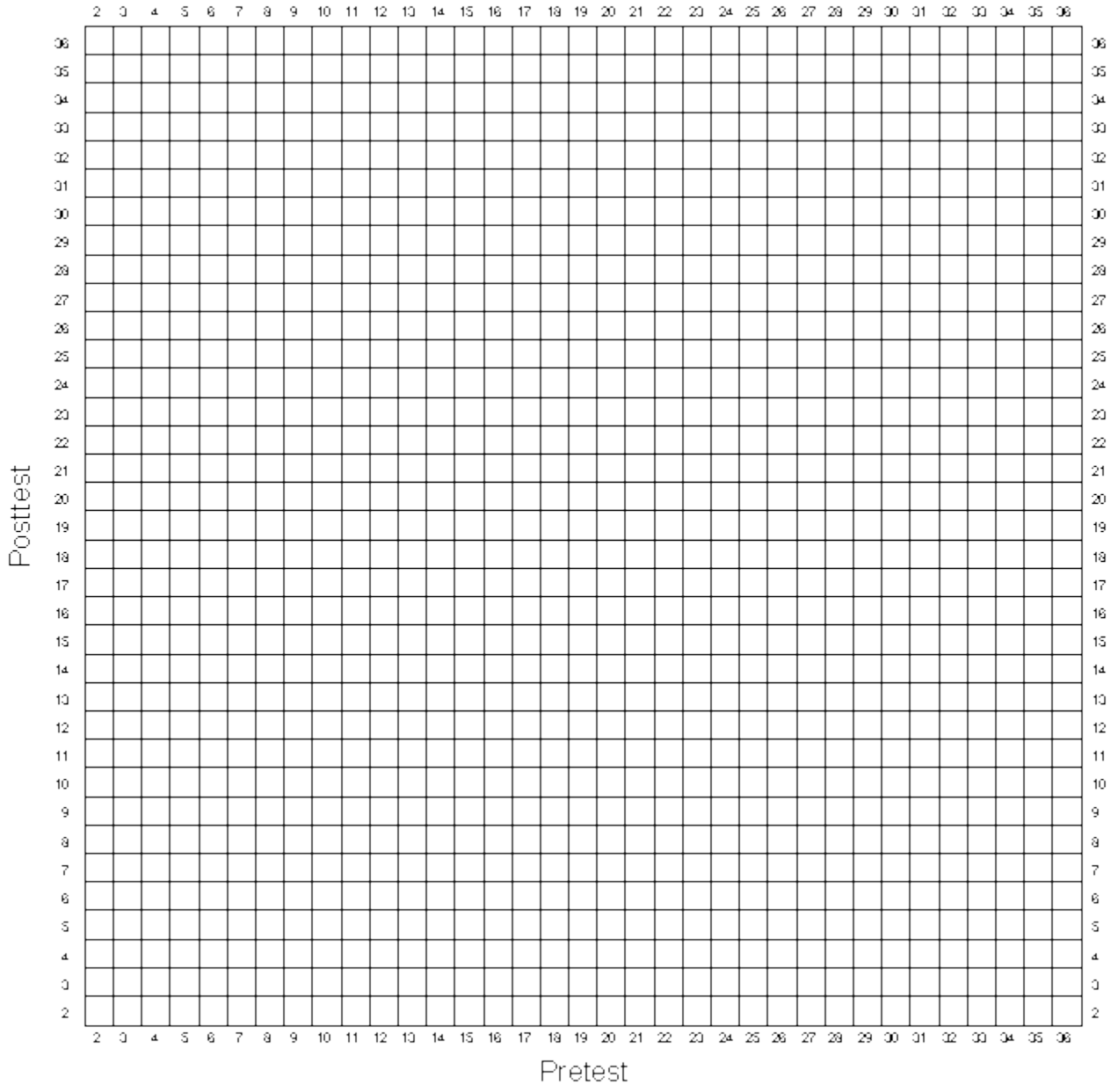
Randomized Experimental Design

Figure 2-2



Randomized Experimental Design

Figure 2-3



The Nonequivalent Group Design

In this exercise you are going to create a nonequivalent group or an untreated control group design of the form

N O X O
N O O

where each O indicates an observation or measure on a group of people, the X indicates the implementation of some treatment or program, separate lines are used to depict the two groups in the study, the N indicates that assignment to either the treatment or control group is not controlled by the researcher (the groups may be naturally formed or persons may self-select the group they are in), and the passage of time is indicated by moving from left to right. We will assume that we are comparing a program and comparison group (instead of two programs or different levels of the same program).

This design has several important characteristics. First, the design has pretest and posttest measures for all participants. Second, the design calls for two groups, one which gets some program or treatment and one which does not (termed the "program" and "comparison" groups respectively). Third, the two groups are nonequivalent, that is, we expect that they may differ prior to the study. Often, nonequivalent groups are simply two intact groups which are convenient to the researcher (e.g., two classrooms, two states, two cities, two mental health centers, etc.).

You will use the pretest and posttest scores from the first exercise as the basis for this exercise. The first thing you need to do is to copy the pretest scores from column 5 of Table 1-1 into column 2 of Table 3-1. Now, you have to divide the 50 participants into two nonequivalent groups. We can do this in several ways, but the simplest would be to consider the first 25 persons as being in the program group and the second 25 as being in the comparison group. The pretest and posttest scores of these 50 participants were formed from random rolls of pairs of dice. Be assured, that on average these two subgroups should have very similar pretest and posttest means. But in this exercise we want to assume that the two groups are nonequivalent and so we will have to make them nonequivalent. The easiest way to make the groups nonequivalent on the pretest is to add some constant value to all the pretest scores for persons in one of the groups. To see how you will do this, look at Table 3-1. You should have already copied the pretest scores (X) for each participant into column 2. Notice that column 3 of Table 3-1 has a number "5" in it for the first 25 participants and a "0" for the second set of 25 persons. These numbers describe the initial pretest differences between these groups (i.e., the groups are nonequivalent on the pretest). To create the pretest scores for this exercise add the pretest scores from column 2 to the constant values in column 3 and place the results in column 4 of Table 3-1 under the heading "Pretest (X) for Nonequivalent Groups". Note that the choice of a difference of 5 points between the groups was arbitrary. Also note that in this simulation we have let the program group have the pretest advantage of 5 points.

Now you need to create posttest scores. You should copy the posttest scores from column 6 of Table 1-1 directly into column 5 of Table 3-1. In this simulation, we will assume that the program has an effect and you will add 7 points to the posttest score of each person in the program group. In Table 3-1, the initial group difference (i.e., 5 points difference) is listed again in column 6 and the program effect or gain (i.e., 7 points) in column 7. Therefore, you get the final posttest score by adding the posttest score from the

first exercise (column 5), the group differences (column 6) and the program effect or gain (column 7). The sum of these three components should be placed in column 8 of Table 3-1 labeled "Posttest Y for Nonequivalent Groups".

It is useful at this point to stop and consider what you have done. When you combine the measurement model from the first exercise with what you have done here, we can represent each personÕs pretest score with the formula

$$X = T + D + e_x$$

where

X = the pretest score for a person

T = the true ability or true score (based on the roll of a pair of dice)

D = initial group difference (D = 5 if the person is in the program group; D = 0 if in comparison group)

e_x = pretest measurement error (based on the roll of a pair of dice)

Similarly, we can now represent the posttest for each person as

$$Y = T + D + G + e_y$$

Y = the posttest score for a person

T = the same true ability as for the pretest

D = the same initial group difference as on the pretest

G = the effect of the program or the Gain (G = 7 for persons in the program; G = 0 for comparison persons)

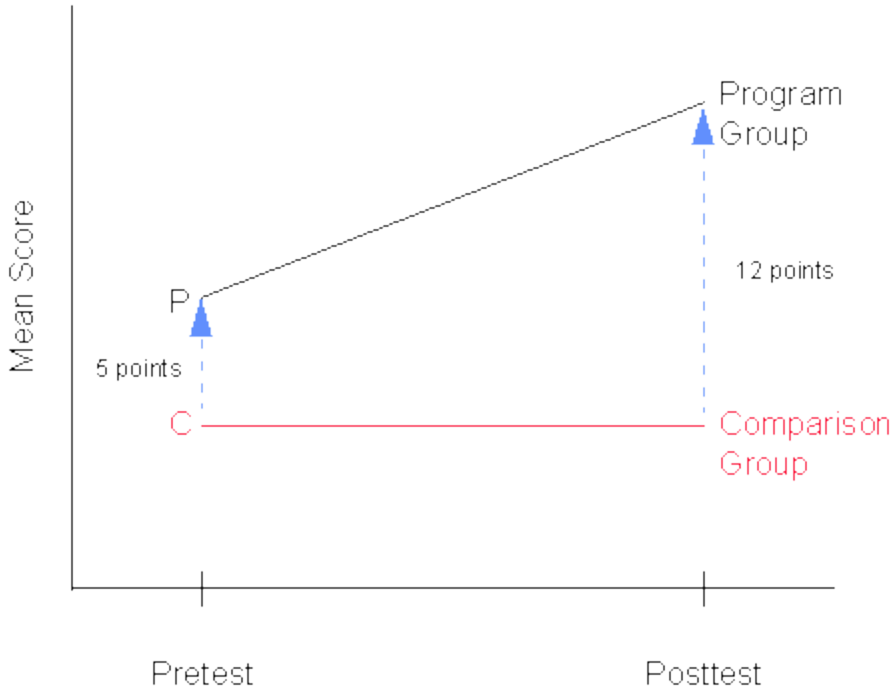
e_y = posttest measurement error (based on a different roll of the dice than pretest error)

It is important to get a visual impression of the data and so, as in the first two exercises, you should graph the univariate and bivariate distributions. Remember that as in the randomized experimental simulation you need to distinguish the program group scores from the comparison group scores on all graphs. Graph the pretest distribution in Figure 3-1, the posttest in Figure 3-2, and the bivariate distribution in Figure 3-3. As before, you should also estimate the central tendency in the univariate distributions, taking care to do this separately for each group. And, you should visually fit a line through the bivariate data, fitting separate lines for the program and comparison groups.

When all of this is completed you should be convinced of the following:

- There are differences between the program and comparison groups on the pretest. If you examine the pretest distributions in Figure 3-1, you should see that the central score for the program group is about 5 points higher than the central score of the comparison group (this is no surprise because you added in the 5 points). This difference is typical of what we expect when we use nonequivalent groups in research and simply tells us that prior to the study one group is higher than the other on the pretest characteristic.

- There are even larger differences between the groups on the posttest. In fact, the posttest difference between groups should be about 12 points (again, this is no surprise because you added in 5 + 7 points). If this were real data and you were going to analyze it, you would probably begin to suspect that your program may have had an effect because the posttest difference exceeds the pretest difference.
- If you were to graph the central values for the pretest and posttest for the two groups, you would probably get a picture that looks something like this:



One alternative explanation (for a program effect) that you would have to consider is the possibility of a selection-maturation threat, that is, that your two groups are maturing at different rates. However, you know this is not the case because you specifically put in the same size group difference of 5 points on both the pretest and posttest (i.e., in the absence of the program, the groups did not mature at different rates). Nevertheless, if you were analyzing data like this in real life, you would have to assume that in the absence of the program the differences between the groups were the same on the pretest and posttest and that any additional difference (in this case 7 points) must be due to the program. You might know from previous research that a maturational pattern like the one in the above figure would be unlikely and rule out the threat as being improbable on that basis. Nevertheless, it should be apparent that you would be much better off, if you had a better idea of how the two groups would have changed from pre to post in the absence of the program. If you had taken an additional pretest observation (i.e. double pretest or the "dry run" experiment), you would have a much better idea of whether selection-maturation is a legitimate threat. In any event, you should be more firmly convinced of the importance of selection bias threats in nonequivalent group designs of this type.

- You should also note what would happen if you analyzed the data in other ways. Obviously a simple t-test of differences on the posttest would give an inappropriately large estimate of program effect -- in this example, it would tell you that the groups differ by about 12 points, but you know that a good deal of that is due to initial differences. On the other hand, an analysis of variance (or t-test) on gain scores would work here but only because you know that without the

program (i.e., if you had not added the 7 point program effect) the two groups would have gained, on the average, exactly the same amount (in this simulation, they would have gained nothing!). You should be convinced then that the analysis of variance on gain scores relies on the assumption of equal gain in both groups in the absence of the program.

- You have only simulated one possible outcome of many. You could, for example, simulate a null case (i.e., no effect of the program) simply by omitting the 7 points added to the program group persons. You could have a constant maturation rate by adding a constant value to all posttest scores. Or, you could simulate a selection-maturation problem by adding different constants to the posttest scores (or true scores) of the two groups. Or you could start out with an inferior program group by adding the group difference to the comparison group instead.
- Finally, you should also recognize an important fact about selection bias which is not illustrated in this exercise. When we select nonequivalent groups we expect that they may differ on one or more characteristics prior to the study. If we find that the pretest scores of our two groups are equal, we cannot assume that there is no selection bias or difference between the groups. The pretest averages could be equal by chance or the groups could differ on any number of other characteristics that are not measured by the pretest but nevertheless affect the posttest scores. We cannot conduct a t-test on pretest differences, find that there is no significant difference and conclude that selection bias is not a problem. Selection bias occurs whenever our groups differ on some pre-study characteristic that affects the posttest and when this pre-study difference is not perfectly described or "accounted for" by the difference on the pretest.

Nonequivalent Group Design
Table 3-1

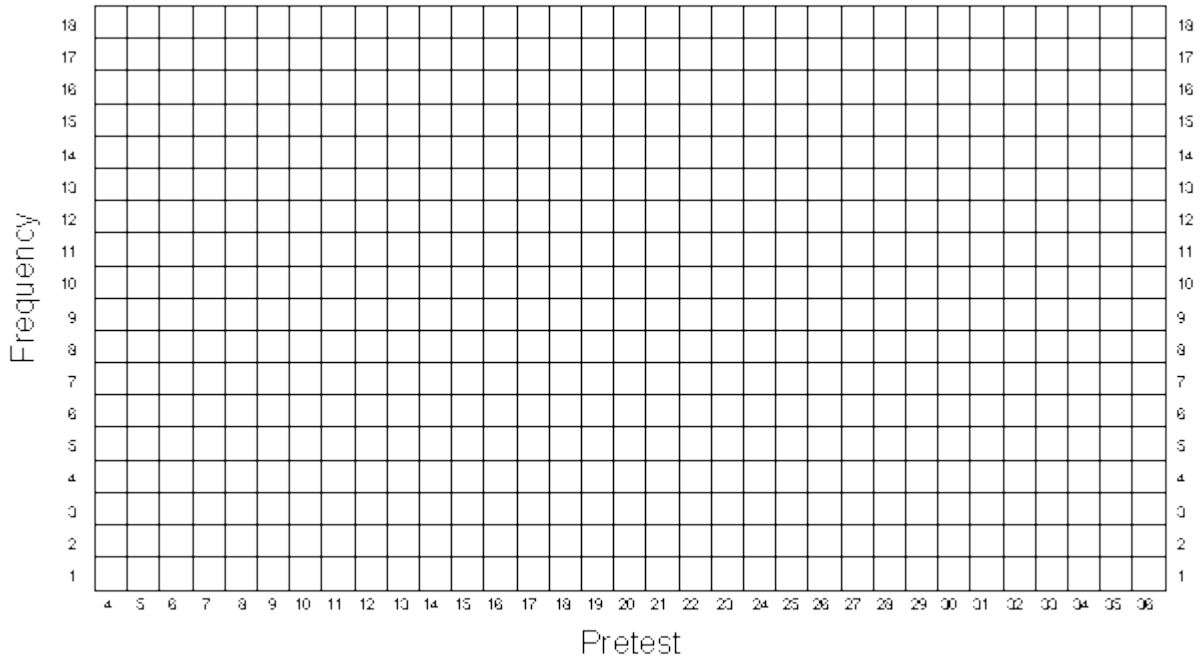
1	2	3	4	5	6	7	8
Person	Pretest X from Table 1-1	Pretest Group Difference	Pretest (X) for Nonequi-valent Groups	Posttest Y from Table 1-1	Posttest Group Difference	Effect of Program (G)	Posttest (Y) for Nonequi-valent Groups
1		5			5	7	
2		5			5	7	
3		5			5	7	
4		5			5	7	
5		5			5	7	
6		5			5	7	
7		5			5	7	
8		5			5	7	
9		5			5	7	
10		5			5	7	
11		5			5	7	
12		5			5	7	
13		5			5	7	
14		5			5	7	
15		5			5	7	
16		5			5	7	
17		5			5	7	
18		5			5	7	
19		5			5	7	
20		5			5	7	
21		5			5	7	
22		5			5	7	
23		5			5	7	
24		5			5	7	
25		5			5	7	

Nonequivalent Group Design
Table 3-1 (cont.)

1	2	3	4	5	6	7	8
Person	Pretest X from Table 1-1	Pretest Group Difference	Pretest (X) for Nonequi-valent Groups	Posttest Y from Table 1-1	Posttest Group Difference	Effect of Program (G)	Posttest (Y) for Nonequi-valent Groups
26		0			0	0	
27		0			0	0	
28		0			0	0	
29		0			0	0	
30		0			0	0	
31		0			0	0	
32		0			0	0	
33		0			0	0	
34		0			0	0	
35		0			0	0	
36		0			0	0	
37		0			0	0	
38		0			0	0	
39		0			0	0	
40		0			0	0	
41		0			0	0	
42		0			0	0	
43		0			0	0	
44		0			0	0	
45		0			0	0	
46		0			0	0	
47		0			0	0	
48		0			0	0	
49		0			0	0	
50		0			0	0	

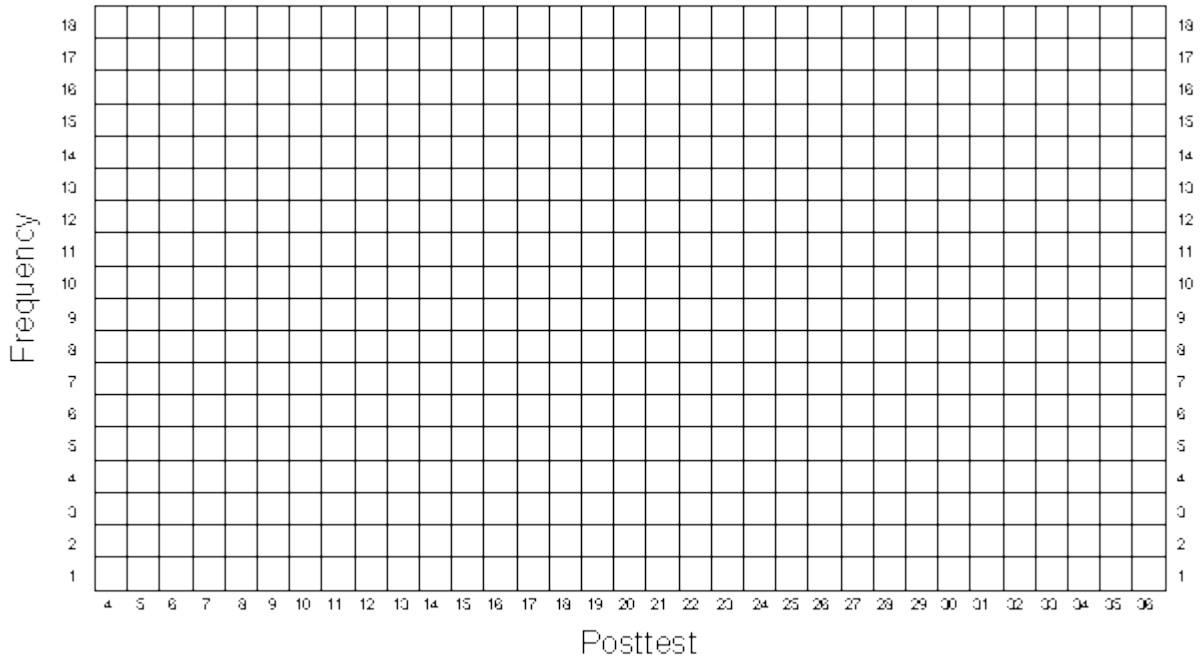
Nonequivalent Group Design

Figure 3-1



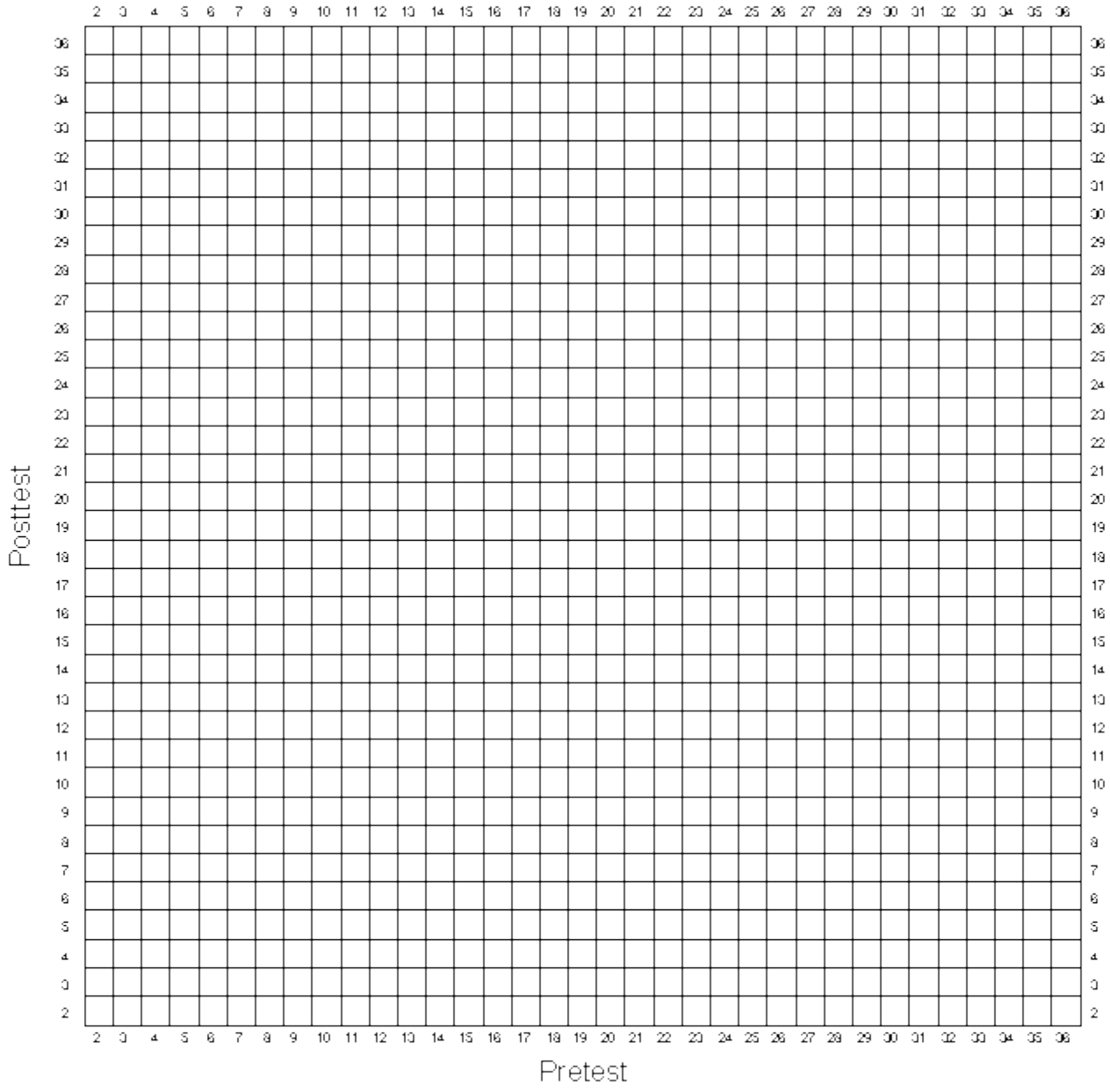
Nonequivalent Group Design

Figure 3-2



Nonequivalent Group Design

Figure 3



The Regression Discontinuity Design

In this exercise you are going to create data for a regression-discontinuity design. It can be depicted in notational form as:

C O X O
C O O

where each O indicates an observation or measure on a group of people, the X indicates the implementation of some treatment or program, separate lines are used to depict the two groups in the study, the C indicates that assignment to either the treatment or control group is done using a cutoff score on the pretest assignment measure, and the passage of time is indicated by moving from left to right. We will assume that we are comparing a program and comparison group (rather than a relative comparison of two programs or different levels of the same program).

The regression-discontinuity design is a type of nonequivalent group design that is characterized by its method of assigning persons to groups using a cutoff score on an assignment measure -- all persons who score above the cutoff are assigned to one group while those scoring on the other side are assigned to the other. Two things need to be decided when selecting a cutoff value. First we need to decide whether the high or low pretest scorers will receive the program. We might give the program to the high pretest scorers if we are studying the effects of scholarships (high achievement), awards (high performance), novel medical treatments or therapies (high on measures of illness) and so on. We might give the program to the low pretest scorers when studying compensatory education (low achievement), poverty (low income), and so on. In this exercise we will simulate a program given to the low pretest scorers. Second, we need to decide the specific value of the pretest cutoff. In the real world, cutoff values are selected in a number of ways. When there are a limited number of program openings, the cutoff score can be selected so that exactly the desired number of persons score either above or below it (depending on whether the program goes to high or low scorers). In other situations, some theoretical value is appropriate for the cutoff. For example, the pretest average might be chosen as the cutoff because in a particular context it makes sense to give the program to those who are "below average" or "above average". In this exercise we arbitrarily use a cutoff equal to the theoretical pretest average.

You will again make use of the pretest and posttest scores that you generated in the first exercise. If you recall that the pretest scores (as generated in the first exercise) can range from 4 to 24, it should be clear that the expected pretest average is 14 units. Thus, we will assign all cases having a pretest score less than or equal to 14 units to the program group and all others to the comparison group (remember that in this simulation the program is given to the low pretest scorers). The assignment strategy can be summarized as follows:

Z = 1 if X <= 14
0 otherwise

where Z is the 0,1 "dummy" assignment variable.

You will generate the data for this exercise using Table 4-1. In the first column of Table 4-1 you should copy the pretest scores (Table 1-1, column 5) from the first exercise into column 2 of Table 4-1. Now, examine the pretest score for person 1. If it is less than or equal to 14, enter a '1' in Column 3 of Table 4-1 labeled Group Assignment (Z). If it is 15 or higher, enter a '0'. Continue doing this for all 50 persons.

When you have finished, notice that the next column, labeled "Hypothetical Program Effect" consists entirely of '7's, that is, the program will increase the posttest scores of each program participant by 7 units. But not everyone gets the program and so not everyone should get the effect of 7 units. You only want those persons who have a Z = 1 (program persons) to get the 7.

An easy way to accomplish this is to multiply the assignment variable (Column 3) by the effect size (Column 4) and put the result in Column 5, labeled "Effect of Program". So, the fourth column should have '7's for all program persons and '0's for all comparison group persons. Next, you should copy the posttest scores from the first exercise (Column 6 of Table 1-1) into column 6 of Table 4-1. Finally, to get the posttest scores with the program effect included you simply add the "Effect of Program" (Column 5) and posttest scores (Column 6) and place the result in Column 7 of Table 4-1 labeled "Posttest (Y) for Regression-Discontinuity Design."

It is useful at this point to stop and consider what you have done. In the first exercise you generated the pretest according to

$$X = T + e_x$$

and in this exercise you constructed the program assignment variable Z using a cutoff rule. Then, using a hypothetical program effect of G = 7 units (G for Gain), you constructed the effect of the program by multiplying GZ. You then copied the posttest from the first exercise and you should recall that it was generated by the model:

$$Y = T + e_Y$$

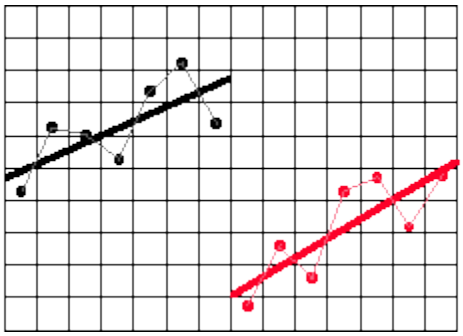
Finally, you added the effect of the program to this posttest value and obtained the posttest for this exercise:

$$Y = T = GZ + e_Y$$

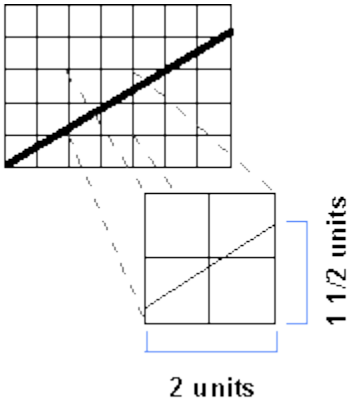
Again, it is always important to examine the data visually and so should graph the univariate pretest distribution in Figure 4-1 and the univariate posttest distribution in Figure 4-2. As in previous exercises, be sure to use a different colored pen or pencil for the program and comparison groups. Also, estimate the central tendency for each group on both graphs using either the counting method or by computing the averages. You should also graph the bivariate distribution as you did before, remembering to keep the marks for the two groups distinct both in color and symbol. Also, estimate the line that fits through the bivariate data.

Let's consider the univariate distributions. Clearly, the pretest distribution in this exercise is identical to the pretest distribution of the first exercise. The only difference is that the program group has scores of 14 or less and the comparison group has scores of 15 or more. Notice that because of this the pretest averages for the two groups are very different. This is what we mean when we say that the regression-discontinuity design induces maximal pretest differences between the groups. Now look at the posttest distribution. If this was all the information you had (i.e., you did not know the pretest information) you would probably conclude that the program and comparison groups don't differ much --that is, the program is not effective. It is only when you consider how different they are on the pretest that you can see there is a program effect, that is, the program group did much better than would have been expected on the basis of their pretest scores.

Next, look at the bivariate distribution. As in the previous exercise, you visually fit separate lines for the program and comparison groups. Let's use these jagged lines to try to estimate a straight line that fits through the data. You will have to do this visually. The figure below shows the dots estimated in each column from a hypothetical example and the lines connecting them. It also shows the straight line that we visually estimated to fit through the jagged one. You should estimate the line for the program and comparison groups separately. The program group line should be to the left of column 14 (and include it) and the comparison group one should be to the right.



You can easily estimate the slopes of these lines. First, take the program group line. Place a dot somewhere on this line at a point where one of the column lines intersects the straight line. Now move exactly two columns to the right and place a dot where the straight line intersects the column line. At this point, you should have something that resembles the following:



You know that the horizontal line is exactly 2 units wide. Measure the vertical distance between the two dots in your graph. Be sure that you measure this distance in terms of the units of the graph. Let's say that you find that it is about 1-1/2 units high. To estimate the slope, you simply construct a ratio where the vertical distance is the numerator and the horizontal distance is the denominator. In this example, you would calculate:

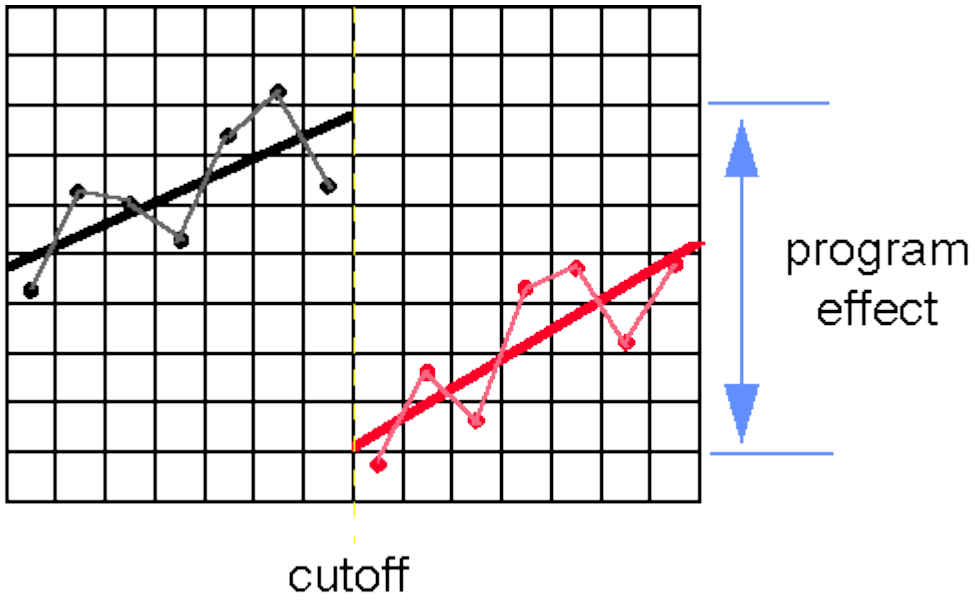
slope = (1-½)/2

= 1.5/2

= .75 or ¾

The slope enables us to say how much change in the vertical direction we get for each 1-unit change in the horizontal direction. In this example, for every increase of 1 unit in the X direction we get an increase of .75 units or 3/4 unit in the Y direction. The estimates of slope for the program and comparison group lines should be very similar.

Now let's estimate the size of the program effect. First, draw a vertical line through the entire bivariate distribution at the cutoff point (i.e., X = 14). Place a dot where the program group straight line intersects the cutoff line. Similarly, place a dot where the comparison group line intersects the cutoff line. Now count the number of vertical units between these two dots. This is the regression-discontinuity estimate of the program effect. You should find that this estimate is about 7 units which is, of course, what you put in. This is illustrated in the figure below.



After completing the previous exercise, you should be convinced of the following:

Although in these dice rolling simulations we have avoided presenting statistical terminology as much as possible, our discussion of the regression-discontinuity design would not be complete without it. After all, the first half of the name of this design is "regression." It should be no surprise that when we statistically analyze this design in the real world, we use regression analysis. Here, we consider some of the major issues involved in such an analysis.

A crucial step in the analysis of data from the regression-discontinuity design involves guessing the true shape of the regression line. In our example this is easy to do because we created the data and we know that the true shape is a straight line in each group. This is because the pretest and posttest both share the same true score. In real life, we don't often know what the true regression shape is, and we have to guess at it. Thus, if you were conducting a real data analysis, you might try a variety of regression lines until you were confident that you had captured this true shape.

Since we know that the true shape in this case is linear, we could construct the appropriate regression model as follows:

$$Y = b_0 + b_1X^* + b_2Z + e_Y$$

where:

Y = the posttest

X* = the pretest minus the cutoff value (i.e., X - 14)

Z = the 0,1 group assignment variable

b_0 = the intercept, that is, the y value at which the comparison group regression line meets the cutoff line

b_1 = the slope (we assume it's the same in both groups)

b_2 = the program effect, that is, the amount you must add or subtract to b_0 in order to find where the program group regression line meets the cutoff line.

e_Y = random error

In regression-discontinuity analysis, we usually subtract the cutoff value from each pretest score before the analysis so that the cutoff is at a value of X = 0 which is the intercept in the model. Notice that the term b_2Z is simply the program effect b_2 times the assignment variable (Z) which is exactly what we put in as GZ. You should be able to estimate all of the b's in the formula above from the bivariate graph. First, b_0, is the posttest value for the point you marked on the cutoff line where the comparison group line intercepts it. Second, b_1, is the estimate of the slope. If your program and comparison group slope estimates differed considerably take the average of the two. Finally, b_2, is the program effect -- the posttest (Y) distance between the two regression lines at the cutoff. Let's say that you estimate $b_0 = 14$, $b_1 = .5$ and $b_2 = 7$. You could then write out the regression formula as:

$$Y = 14 + .5X^* + 7Z$$

(We drop the e_Y term out because that describes deviations from the regression lines.) Basically, when you run a regression-discontinuity analysis you enter in the values for Y, X* (remember to subtract the cutoff from each X) and Z and the regression program gives you the estimates of b_0, b_1, and b_2.

You should be convinced that this single formula describes the regression lines for both groups as well as the program effect. To see this, you can construct the formula for each regression line separately. First, construct the formula for the program group line (substituting your own estimates instead of these) by setting Z = 1 (remember this is the program group):

$$Y_P = 14 + .5X^* + 7(1)$$
$$Y_P = 14 + .5X^* + 7$$
$$Y_P = 21 + .5X^*$$

Now you can construct the formula for the comparison group line by substituting Z = 0.

$$Y_C = 14 + .5X^* + 7(0)$$
$$Y_C = 14 + .5X^*$$

Now, to convince yourself that the program effect is correctly estimated, construct the program effect at the cutoff. Remember that we subtracted the cutoff from each pretest value and so the cutoff is at X* = 0. Therefore, the Y estimate for the program group value at the cutoff in this example would be

$$Y_P = 21 + .5(0)$$
$$Y_P = 21$$

and the comparison group y value at the cutoff would be

$$Y_C = 14 + .5(0)$$
$$Y_C = 14$$

and, therefore, the program effect would be the difference between the two groups or

$$Y_P - Y_C = 21 - 14$$
$$Y_P - Y_C = 7$$

You should get a value close to the value of 7 units which is, of course, what you put in when you constructed the data. It should also be clear that when a dichotomous dummy variable (e.g., Z) is used in a regression equation, you are essentially telling the analysis that you want to fit two lines, one for each group having each value of Z.

Regression-Discontinuity Design
Table 4-1

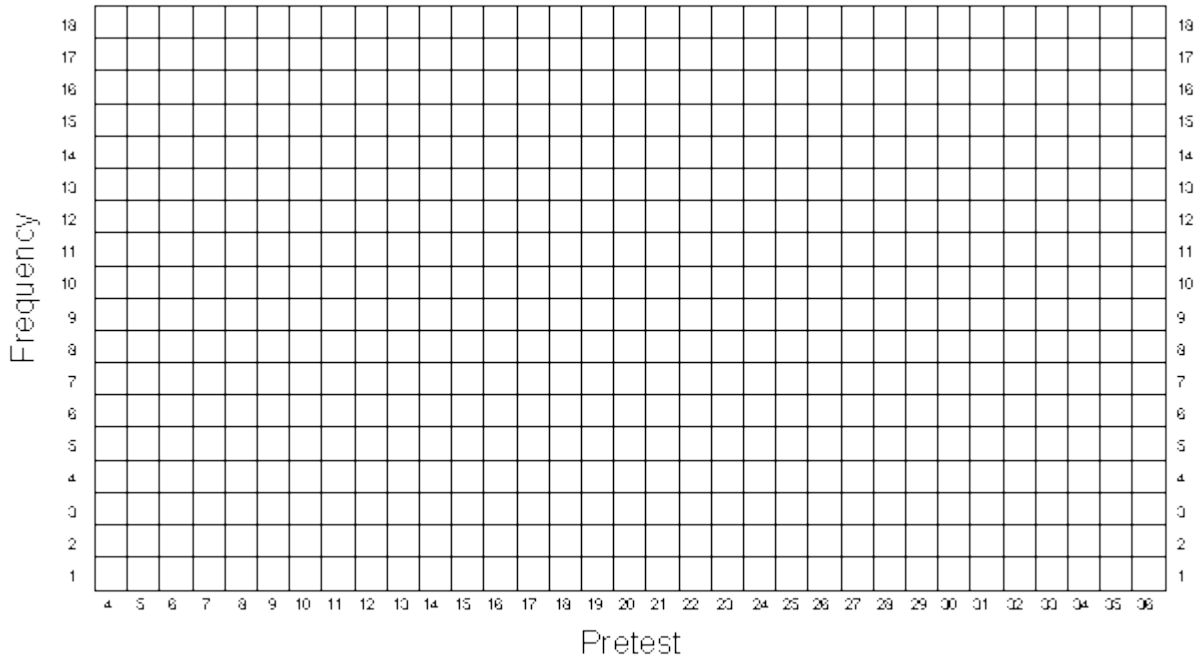
1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Group Assignment Z	Hypothetical Program Effect	Effect of Program	Posttest Y from Table 1-1	Posttest (Y) for Regression-Discontinuity Design
1			7			
2			7			
3			7			
4			7			
5			7			
6			7			
7			7			
8			7			
9			7			
10			7			
11			7			
12			7			
13			7			
14			7			
15			7			
16			7			
17			7			
18			7			
19			7			
20			7			
21			7			
22			7			
23			7			
24			7			
25			7			

Regression-Discontinuity Design
Table 4-1 (cont)

1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Group Assignment Z	Hypothetical Program Effect	Effect of Program	Posttest Y from Table 1-1	Posttest (Y) for Regression-Discontinuity Design
26			7			
27			7			
28			7			
29			7			
30			7			
31			7			
32			7			
33			7			
34			7			
35			7			
36			7			
37			7			
38			7			
39			7			
40			7			
4			7			
42			7			
43			7			
44			7			
45			7			
46			7			
47			7			
48			7			
49			7			
50			7			

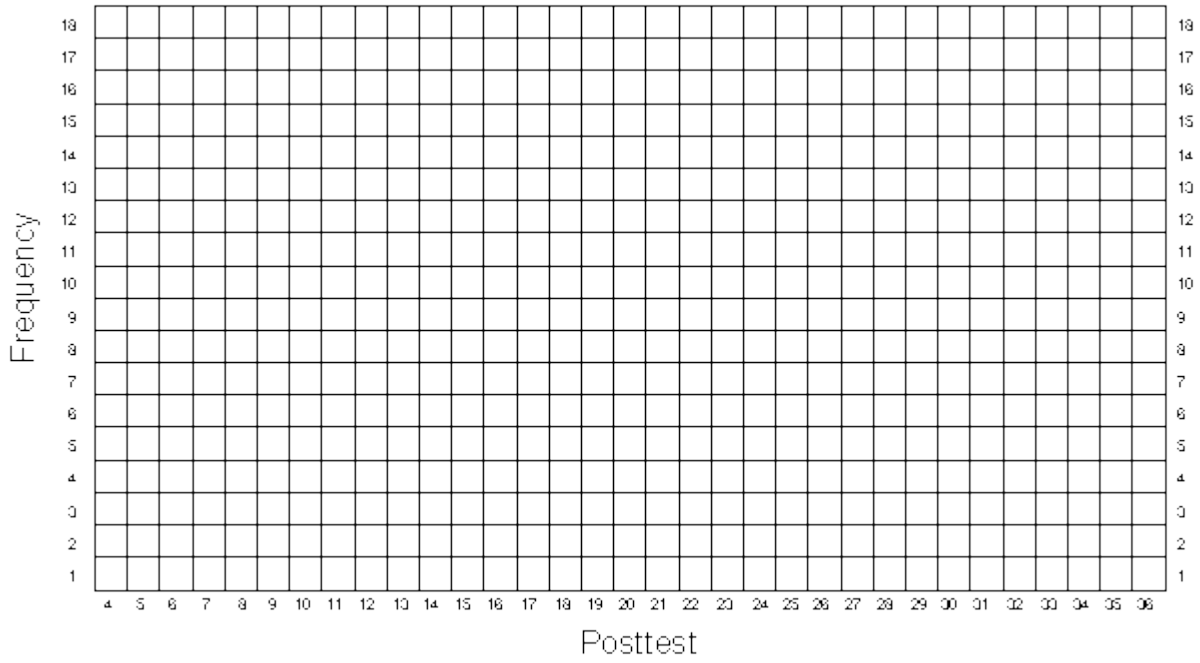
Regression-Discontinuity Design

Figure 4-1



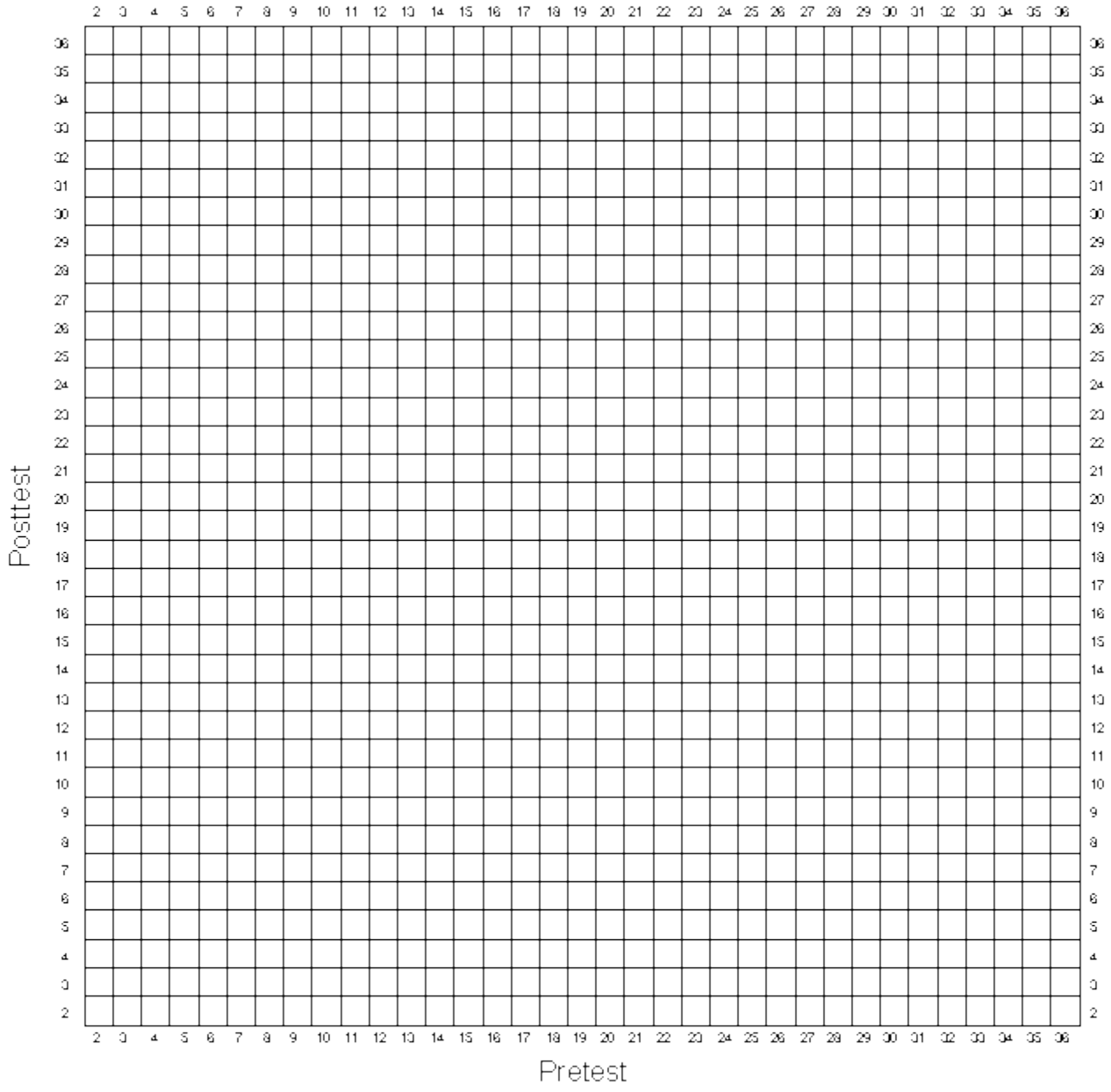
Regression-Discontinuity Design

Figure 4-2



Regression-Discontinuity Design

Figure 4-3



Regression Artifacts

In this exercise we are going to look at the phenomenon of regression artifacts or "regression to the mean." First, you will use the data from the original simulation and create nonequivalent groups just like you did in the Nonequivalent Group Design exercise. Then you will "match" persons from the program and comparison groups who have the same pretest scores, dropping out all persons for whom there is no match. You do this because you are concerned that the groups have different pretest averages, and you would like to obtain "equivalent" groups. Second, you are going to regraph the data for all 50 persons from the Generating Data (GD) exercise, to gain a deeper understanding of regression artifacts.

To begin, review what you did in the NEGD exercise. Starting with 50 pretest and posttest scores (each composed of a common true score and unique error components), you first made the groups nonequivalent on the pretest by adding 5 to each program person's pretest value. This initial difference was the same on the posttest, and so you added the same 5 points there. Finally, you included a program effect of 7 points, added to each program person's posttest score.

In this exercise, you will start with the data in the GD exercise, and will do the same thing you did in the NEGD exercise except that we will not add in a program effect. That is, in this simulation we assume that the program either was never given or did not work (i.e., the null case). The first thing you need to do is to copy the pretest scores from column 5 of Table 1-1 into column 2 of Table 5-1. Now, you have to divide the 50 participants into two nonequivalent groups. We can do this in several ways, but the simplest would be to consider the first 25 persons as being in the program group and the second 25 as being in the comparison group. The pretest and posttest scores of these 50 participants were formed from random rolls of pairs of dice. Be assured, that on average these two subgroups should have very similar pretest and posttest means. But in this exercise we want to assume that the two groups are nonequivalent and so we will have to make them nonequivalent. The easiest way to make the groups nonequivalent on the pretest is to add some constant value to all the pretest scores for persons in one of the groups. To see how you will do this, look at Table 5-1. You should have already copied the pretest scores (X) for each participant into column 2. Notice that column 3 of Table 5-1 has a number "5" in it for the first 25 participants and a "0" for the second set of 25 persons. These numbers describe the initial pretest differences between these groups (i.e., the groups are nonequivalent on the pretest). To create the pretest scores for this exercise, add the pretest scores from column 2 to the constant values in column 3 and place the results in column 4 of Table 5-1 under the heading "Pretest (X) for Regression Artifacts". Note that the choice of a difference of 5 points between the groups was arbitrary. Also note that in this simulation we have let the program group have the pretest advantage of 5 points.

Now you need to create posttest scores. You should copy the posttest scores from column 6 of Table 1-1 directly into column 5 of Table 5-1. In this simulation, we will assume that the program either has no effect or was never given, and so you will not add any points to the posttest score for the effect of the program. But we assume that the initial difference between the groups persists over time, and so you will add to the posttest the 5 points that describes the nonequivalence between groups. In Table 5-1, the initial group difference (i.e., 5 points difference) is listed again in column 6. Therefore, you get the final posttest score by adding the posttest score in column 5 and the group differences in column 6. The sum should be placed in column 7 of Table 5-1 labeled "Posttest Y for Regression Artifacts".

Now, just as you have done in previous exercises, plot the pretest and posttest frequency distributions in Figures 5-1 and 5-2, being sure to use different colors for the program (persons 1-25) and comparison

(persons 26-50) groups. Also, estimate the central tendency for each group on both the pretest and posttest. You should notice that the average of the program groups is about 5 points higher than the average of the comparison group on both measures.

If you were conducting a nonequivalent group design quasi-experiment and obtained the pretest distribution in Figure 5-1, you would rightly be concerned that the two groups differ prior to getting the program. To remedy this, you might think it is a good idea to look for persons in both groups who have similar pretest scores, and use only these matched cases as the program and comparison groups. You might conclude that by only using persons "matched" on the pretest you can obtain "equivalent" groups.

You will match persons on their pretest scores, and put the matched cases in Table 5-2. To do this, first look at the pretest frequency distribution in Figure 5-1. Notice again that the comparison group tended to score lower. Beginning at the lowest pretest score and moving upwards, find the lowest pretest score at which there are both program and comparison persons. Most likely there will be more comparison persons than program ones at the first score that has both. For instance, let's imagine that the pretest score of 9 is the first score that has persons from both groups and that at this value there are two cases from the comparison group and one from the program group. Obviously you will only be able to find one matched pair--you will have to throw out the data from one of the comparison group person because there is only a single program group case available for matching. Since the dice used to generate the data yield random scores, you can simply take the first person in the comparison group (Table 5-1, persons 26-50) who scored a 9 on the pretest. Record that person's ID number in column 1 of Table 5-2, their pretest in column 2 and their posttest score in column 3. Next, find the program person (in Table 5-1, persons 1-25) who also scored a 9 on the pretest and enter that person's ID number in column 4 of Table 5-2, their pretest in column 5 and their posttest score in column 6. Then move to the next highest pretest score in Figure 5-1 for which there are persons from both groups. Again, find matched pairs, and enter them into Table 5-2. Continue doing this until you have obtained all possible matched pairs. Notice that you should never use the same person more than once in Table 5-2.

At this point, you have created two groups matched on the pretest. To do so, you had to eliminate persons from the original sample of 50 for whom no pretest matches were available. You may now be convinced that you have indeed created "equivalent" groups. To confirm this, you might calculate the pretest averages of the program and comparison groups. They should be identical.

Have you in fact, created "equivalent" groups? Have you removed the selection bias (of 5 points) by matching on the pretest? Remember that you have not added in a program effect in this exercise. If you successfully removed the selection difference on the pretest by matching, you should find no difference between the two groups on the posttest (because you only put in the selection difference between the two groups on the posttest). Calculate the posttest averages for the program and comparison groups in Table 5-2. What do you find?

Most of you will find that on the posttest the program group scored higher on average than the comparison group did. If you were conducting this study, you might conclude that although the matched groups start out with equal pretest averages, they differ on the posttest. In fact, you would be tempted to conclude that the program is successful because the program group scored higher than the comparison group on the posttest. But something is obviously wrong here--you never put in a program effect! Therefore, the posttest difference that you are finding must be wrong.

To discover what is wrong you will plot the data in Table 5-2 in a new way. Look at Figure 5-4 labeled "Pair-Link Diagram". Starting with only the comparison persons in Table 5-2, draw a straight line between the pretest and posttest scores of each person. Do the lines tend to go up, down, or stay the same from pretest to posttest? Next, using a different colored pen, draw the lines for the program group persons in Table 5-2. In which direction do these lines go? You should find that most of the program group lines go down while most of the comparison group lines go up from pretest to posttest. As a result of what you have seen, you should be convinced of the following:

- The average posttest difference between the program and comparison group is entirely due to regression artifacts that result from the matching procedure. Recall that because of the pretest difference of 5 points, which you put in, the entire program group had a higher pretest average than the entire comparison group. When you matched persons on the pretest, you were actually selecting the higher scoring comparison persons and the lower scoring program persons. Therefore, we expect the matched comparison group to regress down toward the entire group's mean and the matched program group to regress up toward the entire group's mean.
- In this simulation you made the program group higher on the pretest by adding 5 points. You should recognize that if the comparison group had been given this initial "advantage" the results of matching would have been reversed. In this case the matched comparison group would have had a higher posttest average than the matched program group. You would mistakenly conclude that the program was harmful--that is, even though the two matched groups start with equal pretest averages, the program group loses relative to the comparison group. Of course, any gain or loss is due to regression artifacts which result from a matching process that selects persons from the higher end of the distribution in one group and the lower end in the other.
- Matching should not be confused with blocking. If you had taken persons from two groups which differ on the pretest, matched them on pretest scores and then randomly assigned one of each pair to the program and comparison group, you would have equal numbers of advantaged and disadvantaged persons in each group. In this case, regression artifacts would cancel out and would not affect results.

Why do regression artifacts occur? We can get some idea by looking at a pair-link diagram for the entire set of 50 persons in the original Generating Data exercise. Draw the pair-links for each of the 50 persons of Table 1-1 on Figure 5-5. Recall that for this original set of data we had only one group (i.e., no program and comparison group), no selection biases and no program effects. You should be convinced of the following:

- Persons who score extremely high or extremely low on the pretest seldom do as extremely on the posttest. That is, there should be very few pair-link lines which go from a low pretest score to an equally low posttest score or which go from a high pretest score to an equally high posttest score.
- Recall that the pretest and posttest consists of two components, a true score which is the same on both tests and separate error scores for each. You should know that the regression artifact cannot be due to the true score. If you were to draw a pair-link diagram between the pretest and posttest true score, you would obtain nothing but horizontal lines (no regression) because it is the same for both tests. However, if you drew a pair-link diagram between the pretest error score and the posttest error score, you would see a clear regression effect. People with low pretest errors would tend to have higher posttest error scores and vice versa. This is because the pretest and posttest error scores were based on independent dice rolls, the two sets of error

scores are random or uncorrelated. We can conclude that regression artifacts must be due to the error in measurement, not to the true scores.

- We can also view this in terms of correlations. First, assume that we have no measurement error-- persons always get the same score on the pretest and posttest. In this case, the pair-link diagram would only have horizontal lines, as stated above, and there would be no regression artifact. Furthermore, if people scored the exact same on both tests, there would be a perfect correlation between the two tests (i.e., r = 1). Next, assume that our pretest and posttest are terrible measures that only reflect error (i.e., they do not measure true ability, but do reflect random errors, at two points in time). Here, the two tests would be random or uncorrelated (i.e., r = 0). and we would expect maximum regression to the mean (i.e., no matter what subgroup you select on the pretest, the posttest average of that subgroup will always tend to equal the posttest average of the entire group). You should recognize that the more measurement error you have in the measures, the lower the correlation between the measures. Finally, you should also see that the lower the correlation between two measures the greater the regression artifact and, the higher the correlation the lower the regression.
- Finally, you should recognize that regression artifacts are purely a statistical phenomenon that results from a symmetric subgroup selection and imperfect correlation. This means that when we select a subgroup from the extreme of a distribution, we will find regression to the mean on any variable that is not perfectly correlated with the selection measure. This can lead the unwary analyst to some bizarre conclusions. For example, let us say you wanted to look at the effect of a special educational program that was given to all students in a school. Assume that you have pretest and posttest scores for everyone (but there is no control group). You would like to know whether subgroups in the school improved. First, you look at the students who scored low on the pretest. They appear to improve on the posttest (regression artifacts, of course). Next, you look at the students who scored high on the pretest. They appear to lose ground on the posttest. You might incorrectly conclude the education helps low scoring students but hurts high scoring students. Now let us say you decide to look at groups who differ on the posttest. The low posttest scorers did much better on the pretest. The high posttest scorers did much worse on the pretest. It almost appears as if students regress backwards in time. But by now you should recognize that this is simply a regression artifact that results from selection of groups on the extremes of the posttest and the imperfect correlation between the pretest and posttest.

Regression Artifacts
Table 5-1

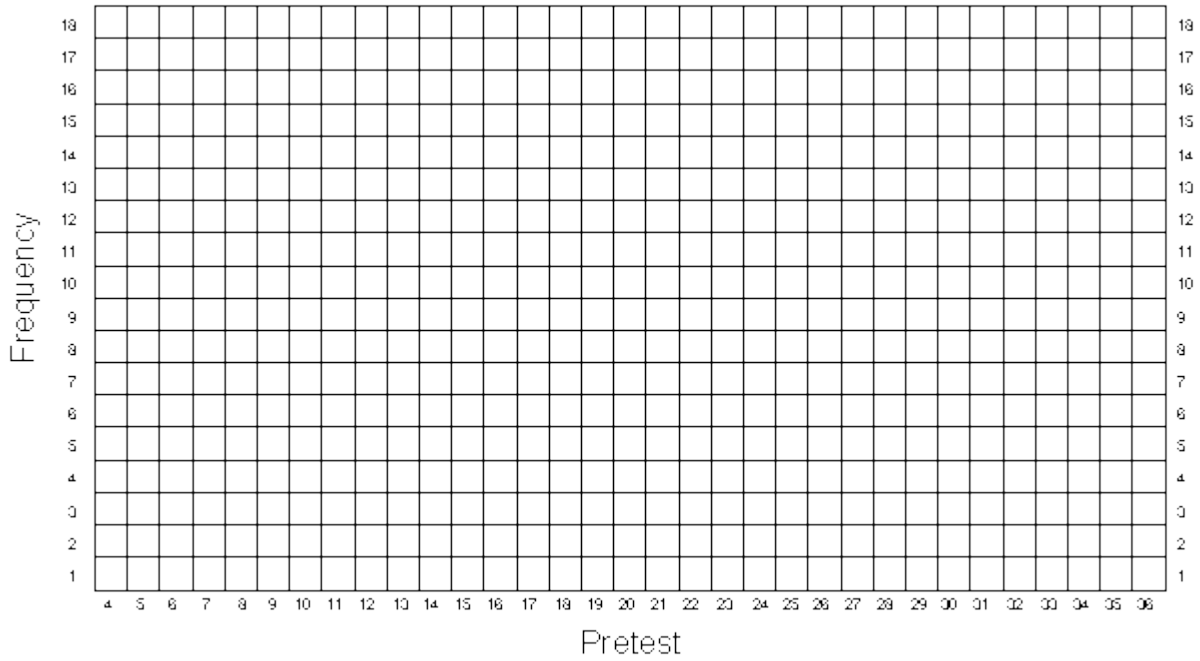
1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Pretest Group Difference	Pretest (X) for Regression Artifacts	Posttest Y from Table 1-1	Posttest Group Difference	Posttest (Y) for Regression Artifacts
1		5			5	
2		5			5	
3		5			5	
4		5			5	
5		5			5	
6		5			5	
7		5			5	
8		5			5	
9		5			5	
10		5			5	
11		5			5	
12		5			5	
13		5			5	
14		5			5	
15		5			5	
16		5			5	
17		5			5	
18		5			5	
19		5			5	
20		5			5	
21		5			5	
22		5			5	
23		5			5	
24		5			5	
25		5			5	

Regression Artifacts
Table 5-1 (cont.)

1	2	3	4	5	6	7
Person	Pretest X from Table 1-1	Pretest Group Difference	Pretest (X) for Regression Artifacts	Posttest Y from Table 1-1	Posttest Group Difference	Posttest (Y) for Regression Artifacts
26		0			0	
27		0			0	
28		0			0	
29		0			0	
30		0			0	
31		0			0	
32		0			0	
33		0			0	
34		0			0	
35		0			0	
36		0			0	
37		0			0	
38		0			0	
39		0			0	
40		0			0	
41		0			0	
42		0			0	
43		0			0	
44		0			0	
45		0			0	
46		0			0	
47		0			0	
48		0			0	
49		0			0	
50		0			0	

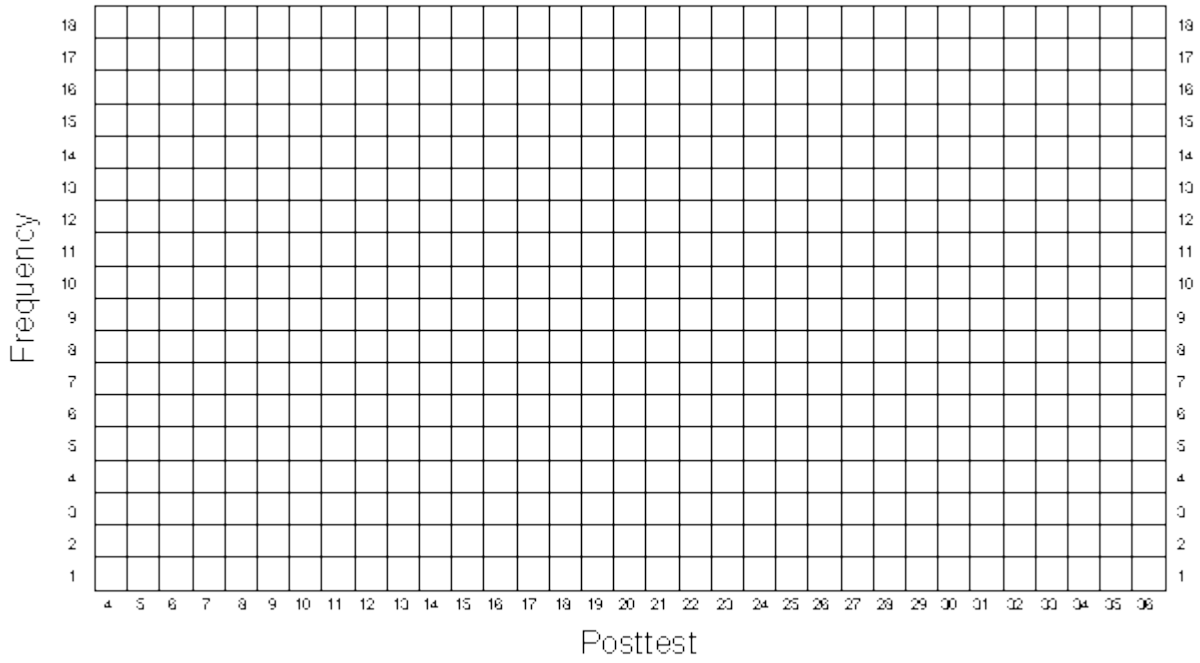
Regression Artifacts

Figure 5-1



Regression Artifacts

Figure 5-2



Regression Artifacts

Figure 5-3

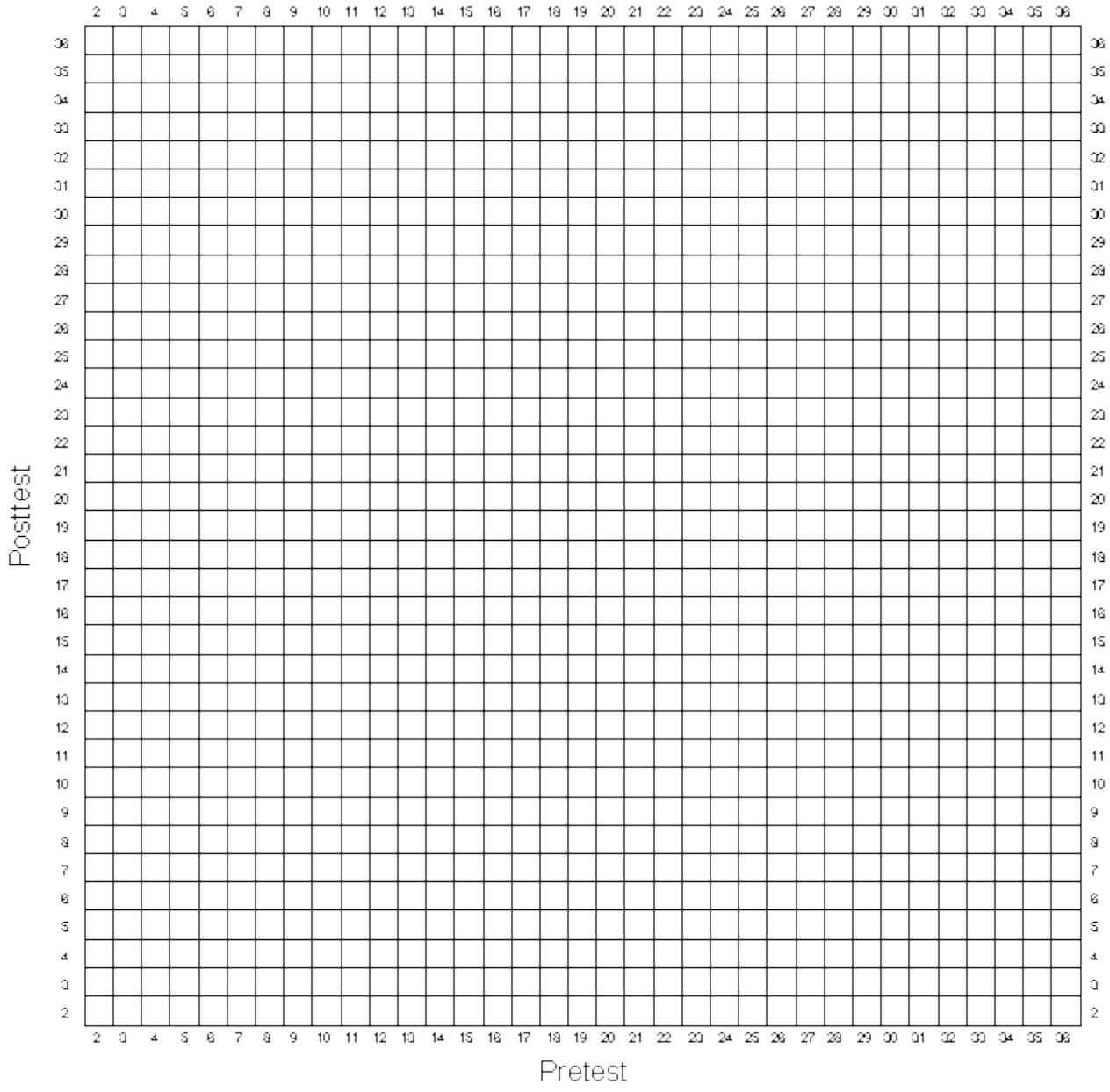
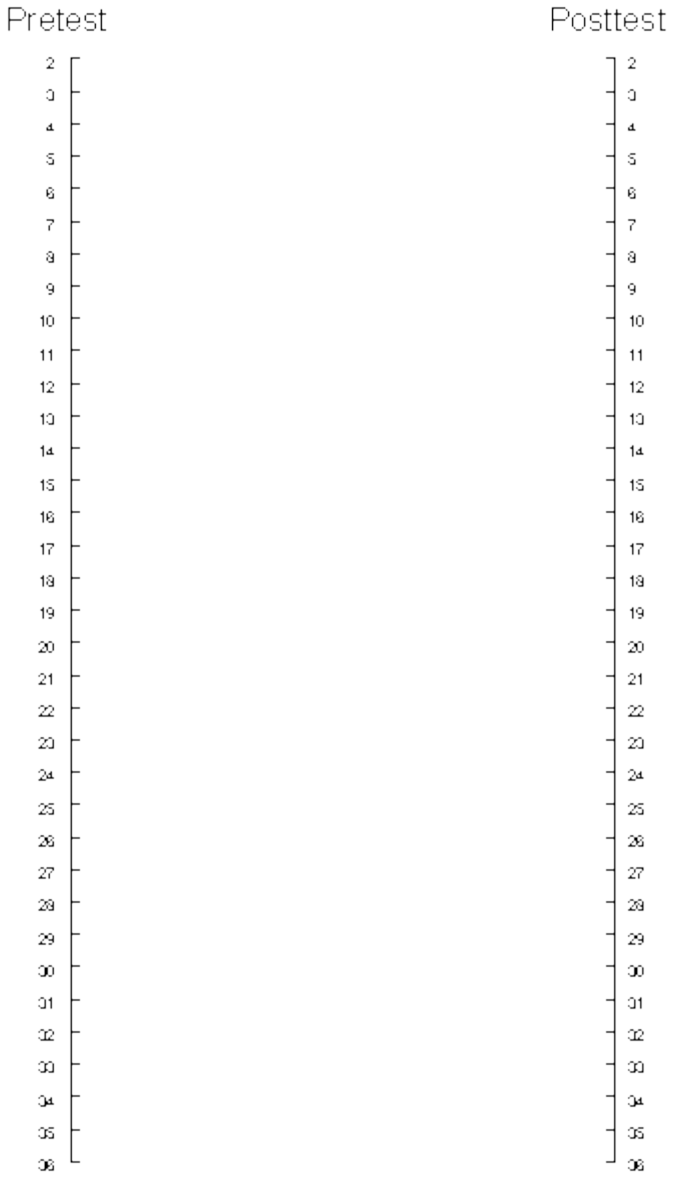


Table 5-2
Matched Cases from Table 5-1

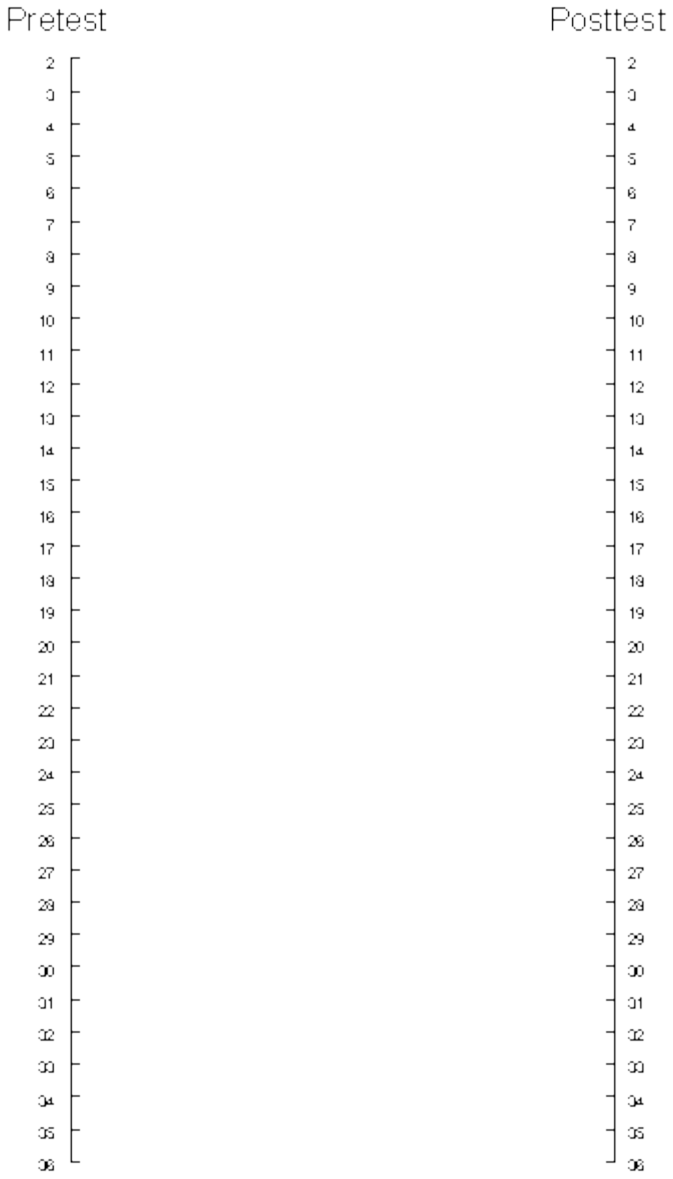
1	2	3	4	5	6
Comparison Group Person Number	Pretest X from Table 5-1	Posttest Y from Table 5-1	Program Group Person Number	Pretest X from Table 5-1	Posttest Y from Table 5-1

	Comparison Group Pretest Average=	Comparison Group Posttest Average=		Program Group Pretest Average=	Program Group Posttest Average=

Regression Artifacts
Figure 5-4
Pair-Link Diagram



Regression Artifacts
Figure 5-5
Pair-Link Diagram



Computer Simulations

The computer exercises that follow utilize a computer program known as MINITAB. It has several distinct advantages for simulation exercises. This statistical package is widely available and relatively inexpensive. The exercises shown here were tested under Release 10.2 of MINITAB for Windows and should run with that or any later release on either Macintosh, IBM-compatible, or even mainframe computers. It is highly interactive -- you enter a command and can immediately see the results. But it can also be run in "batch" mode which allows you to enter a long sequence of commands and have them run at once as a program. Batch programs can also be put in a loop that allows you to run the same sequence of commands many times, accumulating the results from each run. The exercises in this workbook are all set up for interactive mode because that is the best way to learn about computer simulation (see Trochim and Davis (1986) for more details on running programs in batch mode). MINITAB is also very easy to learn. There are a number of readily available instruction manuals and tutorials that introduce the novice to statistical computing.

But MINITAB is not the only computer program that can be used for simulation, nor is it the program of choice for professional simulators. Virtually any statistical package -- SPSS, SAS, SYSTAT, Statview, DataDesk -- can be used for simulations, and each has some advantages and disadvantages. For all its strengths as a teaching tool, MINITAB is not often viewed as a serious tool for advanced statistical analysis in social research because it is slower than others, lacks many of the features of advanced packages, and may not be as precise. However, for the kinds of exercises described in this workbook, MINITAB is an excellent choice.

In order to do these exercises, all you need to know is how to install and start the MINITAB program. This information can be found in the manual that comes with the program. The exercises are "machine independent" -- we do not describe them in terms of any particular operating system or machine configuration. The exercises don't assume any prior knowledge or use of the MINITAB language, although that would be extremely helpful. We encourage you to work through the tutorial in the manual that comes with the program.

We hope that the exercises given here will provide you with a solid and interesting introduction to computer simulation for social research. We believe that you will find that simulation is an important tool for increasing you understanding of social research methodology.

Generating Data

Introduction

This first computer exercise introduces you to many of the basic computer and statistical concepts that will be used in later exercises. In this exercise you are going to create some data and then perform some simple analyses of it. If you follow the steps outlined below you should be able to work through the entire exercise without a problem. However, in order to really benefit, it is important that you work slowly and think about what you are doing at each step. After completing the exercise you are certainly encouraged to play with variations of your own and some that will be suggested. You can ask for a short or long description of any command by typing HINT or HELP followed by the command name. When you encounter a new command you are encouraged to do this. You might also take a moment to see if the command is listed in the MINITAB Handbook.

Now, get into the MINITAB program. If you don't know how to do this, you have to look it up in the MINITAB manual that came with the program. You should see the MINITAB prompt (which looks like this MTB>). Now you are ready to enter the following commands:

```
MTB> Random 10 C1;
SUBC> Normal 0 1.
```

You begin by having the computer create or generate 10 random numbers. We want these numbers to be normally distributed (i.e., to come from a "bell-shaped" distribution) with a mean or average of zero and a standard deviation of 1. (Remember that the standard deviation is a measure of the "spread" of scores around the mean). You told the computer to put these ten numbers in variable Cl. To get an idea of what the RANDOM command does type:

```
MTB> Help Random
```

Before getting to more serious matters, you should play a little with the ten observations you created. First, print them out to your screen.....

```
MTB> Print C1.
```

Or get means and standard deviations....

```
MTB> Describe C1
```

The mean should be near zero and the standard deviation near one. Or, draw a histogram or bar graph....

```
MTB> Histogram C1
```

Does it look like a bell-shaped curve? Probably not, because you are only dealing with 10 observations. Why don't you start over, this time generating 50 numbers instead of 10.... MTB> Random 50 C1;

```
SUBC> Normal 0 1.
```

Notice that you have erased or overwritten the original 10 observations in Cl. If you do....

```
MTB> Print C1
```

all you see are the newest 50 numbers. To describe the data...

```
MTB> Describe C1
```

Notice that the mean and standard deviation are probably closer to 0 and 1 than was the case with 10 observations. Why?

```
MTB> Histogram C1
```

This should look a little more like a normal curve than the first time (although it may still look pretty bizarre).

Simulation: Generation of Two Variables

The above commands were included to familiarize you with the Random/Normal command. Now you will conduct a real simulation. You'll create data according to a simple measurement model. You will generate two imaginary test scores for 500 individuals. If you like, you can imagine that you have given two achievement tests to 500 school children.

The measurement model we'll use assumes that a test score is made up of two parts - true ability and random error. We can depict the model as:

$$O = T + e_o$$

Here, O is the observed score on a test, T is true ability on that test and eo is random error.

Notice what we're doing here. We will create 500 test scores or Os for two separate tests. In real life this is all we would be given and we would assume that each test score is a reflection of some true ability and random error. We would not (in real life) see the two components on the right side of the equation - we only see the observed score. We'll call our first test the X achievement test, or just plain X. It has the model....

$$X = T + e_X$$

which just says that the X test score is assumed to have both true ability and error in measurement. Similarly, we'll call the second test the Y achievement test, or Y, and assume the model....

$$Y = T + e_Y$$

Notice that both of our tests are measuring the same construct, for example, achievement. For any given child, we assume this true ability is the same on both tests (i.e., T). Further, we assume that a child gets different scores on X and Y entirely because of the random error on either tests - if the tests both measured achievement perfectly (i.e., without error) both tests would yield the same score for every child. OK, now try the following....

```
MTB> Random 500 C1;
SUBC> Normal 0 3.
```

```
MTB> Random 500 C2;
SUBC> Normal 0 1.
MTB> Random 500 C3;
SUBC> Normal 0 1.
```

Be sure to enter these exactly as shown. The first command created 500 numbers which we'll call the true scores or T for the 500 imaginary students. The second command generated the 500 random errors for the X test while the final command generated the 500 errors for the Y test. All three (Cl-C3) will have a mean near zero and the true score will have a bigger standard deviation than the two random errors. How do we know that this will be the case? We set it up this way because we wanted to create an X and Y test that were fairly accurate - reflected more true ability than error. Now, name the three variables so you can keep track of them.

```
MTB> Name C1 "true" C2 "x-error" C3 "y-error"
```

Now get descriptive statistics for these three variables....

```
MTB> Describe C1-C3
```

Note that the means and standard deviations should be close to what you specified. Now construct the X test...

```
MTB> Add C1 C2 C4.
```

Remember, Cl is the true score and C2 is random error on the X test. You are actually creating 500 new scores by adding together a true score, Cl, and random error, C2. Now, construct the Y test...

```
MTB> Add C1 C3 C5.
```

Notice that you use the same true ability, Cl (both tests are assumed to measure the same thing) but a different random error.

It would be worth stopping at this point to think about what you have done. You have been creating imaginary test scores. You have constructed two tests which you labeled X and Y. Both of these imaginary tests measure the same trait because both of them share the same true score. This true score (Cl) reflects the true ability of each child on an imaginary achievement test, for example. In addition, each test has its own random error (C2 for X and C3 for Y). This random error reflects all the situational factors (e.g., bad lighting, not enough sleep the night before, noise in the testing room, lucky guesses, etc.) that can cause a child to score better or worse on the test than his true ability alone would yield. One more word about the scores. Because the true score and error variables were constructed to all have zero means, it should be obvious that the X and Y tests will also have means near zero. This might seem like an unusual kind of test score, but it was done for technical reasons. If you feel more comfortable doing so, you may think of these scores as achievement test scores where a positive value indicates a child who scores above average for his/her age or grade, and a negative score indicates a child who scores below average.

If this were real life, of course, you would not be constructing test scores like this. Instead, you would measure the two sets of scores, X and Y, and would do an analysis of them. You would assume that the two measures have a common true score and independent errors, but you would not see these. Thus, you have generated what we call simulated data. The advantage of using such data is that, unlike with real data, you know how the X and Y tests are constructed because you constructed them. You will see

in later simulations that this enables you to test different analysis approaches to see if they give back the results that you put into the data. If the analyses work on simulated data then you might assume that they will also work for real data if the real data meet the assumptions of the measurement model used in the simulations.

Now, pretend that you didn't create the X and Y tests but, rather, that you were given these two sets of test scores and asked to do a simple analysis of them. You might begin by exploring the data to see what it looks like. First, name the two tests....

```
MTB> Name C4 "X" C5 "Y"
```

Try this command...

```
MTB> Info
```

This just tells you how many variables, what names (if any) and how many observations you have. Now, describe the data....

```
MTB> Describe C4-C5
```

By the way, you might also try some of the other column operations listed in MINITAB Help. For example....

```
MTB> Count C4
```

tells you there are 500 observations in C4,

```
MTB> Sum C4
```

gives the sum,

```
MTB> Average C4
```

gives the mean (which should be near zero),

```
MTB> Medi C4
```

gives the median,

```
MTB> Standard C4
```

gives the standard deviation, and

```
MTB> Maxi C4
MTB> Mini C4
```

give the highest and lowest value in C4.

Now look at the distributions....

```
MTB> Histogram C4
MTB> Histogram C5
```

These should look a lot more like bell-shaped or normal curves than the earlier graphs did. Look at the bivariate relationship between X and Y....

```
MTB> Plot C5 * C4; SUBC> symbol.
```

Notice a few things. You plotted C5 on the vertical axis and C4 on the horizontal. Each point on the graph indicates an X score paired with a Y score. It should be clear that the X and Y tests are positively correlated, that is, higher scores on one test tend to be associated with higher scores on the other. To confirm this, do...

```
MTB> Correlation C4 C5
```

The correlation should be near .90. You can predict scores on one test using scores on the other. To do this you will use regression analysis. Fit the straight-line regression of Y on X...

```
MTB>Regress;
SUBC> Response 'Y';
SUBC> Continuous 'X';
SUBC> Terms 'X'.
```

For now, don't worry about what all the output means (although you might want to start looking at the section on Regression in MINITAB Help). The regression equation describes the best-fitting straight line for the regression of Y on X. You could draw this line on the graph you did earlier. Just substitute some values in for X (try X = O, 1, -1, 2, and -2) and calculate the Y using the equation which the regression analysis gives you. Then plot the X, Y, pairs and you will see that they fall on a straight line. Recall from your high school algebra days that the number immediately to the right of the equal sign is the intercept and tells you where the line hits the Y axis (i.e., when x = 0). The number next to the X variable name is the slope. It is possible to look at a plot of the residuals and the regression line if we use the subcommand form of the regress statement....

```
MTB> Regress;
SUBC> Response 'Y';
SUBC> Continuous 'X';
SUBC> Terms 'X';
SUBC> Residuals C20;
SUBC> Coefficients C22.
MTB> Let C21=C5-C20
```

We have arbitrarily chosen columns C20-C22 to store the residuals, predicted values and coefficients, respectively. The predicted Y value is simply the observed Y minus the residual. The LET command is used to construct the predicted Y value. (Try 'Help regress' to get information about the command). Now, to do a plot of the regression line you plot the predicted values against the X variable....

```
MTB> Plot C21 * C4;
SUBC> symbol.
```

This is actually a plot of the straight line that you fit with the regression analysis. It doesn't look like a "perfect" straight line because it is done on a line printer and there is rounding error, but it should give you some idea of the type of line that you fit. Now, you can also look at the residuals (i.e., the Y-distance from the fitted regression line to each of the data points). To do this type....

```
MTB> Plot C20 * C4;
SUBC> symbol.
```

Notice that the bivariate distribution is circular in shape indicating that the residuals are uncorrelated with the X variable (remember the assumption in regression that these must be uncorrelated?). This graph shows that the regression line fits the data well - there appear to be about as many residuals which are positive (i.e., above the regression line) as negative You might also want to examine the assumption that the residuals are normally distributed. Can you figure out a way to do this?

Now, you should again stop to consider what you have done. In the first part of the exercise you generated two imaginary tests, X and Y. In the second part you did some analyses of these tests. The analyses told you that the means of the tests were near zero, which is no surprise because that's the way you set things up. Similarly, the bivariate graph and the correlation showed you that the two tests were positively related to each other. Again, you set them up to be correlated by including the same true ability score in both tests. Thus, in this first simulation exercise, you have confirmed through simulation that these statistical procedures do tell you something about what is in the data.

It would probably be worth your time to play around with variations on this exercise. This would help familiarize you with MINITAB and with basic simulation ideas. For example, try some of the following....

Change the reliability of the X and Y tests. Recall that reliability is simply the ratio of true score variance to total score variance. You can increase reliability by increasing the standard deviation in the first Random/Normal statement from 3 to some higher number (while leaving the error variable standard deviations at 1). Similarly, you can lower the reliability by lowering the true score standard deviation to some value less than 3. Look at what happens to the correlation between X and Y when you do this. Also, look at what happens to the slope estimate in the regression equation.

Construct tests with more "realistic" averages. You can do this very simply by putting in a different mean than 0 in the first Random/Normal statement. However, you should note that the true score measurement model always assumes that the mean of the error variables is zero. Confirm that the statistical analyses can detect the mean that you put in.

You can always generate more than two tests. Just make sure that each test has some element of true score and error. If you really want to get fancy, why not try generating three variables using two different true scores. Have one variable get the first true score, one get only the second true score, and one have both true scores (you'll have to add in both in a Let statement). Of course, all three variables should have their own independent errors. Look at the correlations between the three variables. Also, try to run a regression analysis with all three (look at the Chapter in the MINITAB Handbook or try the Help command to see how to do this).

One concept that we will discuss later in the simulation on nonequivalent group designs involves what happens in a regression analysis when we have error in the independent variable. To begin exploring this idea you might want to rerun the simulation with the one difference being that you don't add in the error on the X test (i.e., when you do the Let command the first time you leave out the C2 variable). Here, the X measure would have nothing but true score - it would be a perfect measure of true ability. See what happens to the correlation between X and Y. Also, see how the slope in the regression differs from the slope in the original analysis. Now, run the simulation again, this time including the error in the X test but not in the Y test. Again, observe what happens to the correlation and regression coefficient. The key question is whether there is bias in the regression analysis. How similar are the results from the

three simulations (i.e., the original simulation, the perfect X simulation and the perfect Y simulation)? The hard question is why the results of the three simulations come out the way they do. If you are concerned that the results differ because the random numbers you generate are different, run the original three Random/Normal commands, do the original simulation as is, and then run the X-perfect and Y- perfect simulations beginning with the let commands and eliminating either the X or Y error terms. In this case you are using the same set of random numbers for all three runs and differences in results can be attributed to use of either perfect or imperfect measurement. At any rate, this is a complex issue that will be discussed in more detail later.

One of the best ways to learn MINITAB is to do it. Don't hesitate to sit at the computer and use the Help and Hint commands to explore the system.

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

```
Random 10 C1;
Normal 0 1.
Print C1.
Describe C1
Histogram C1
Random 50 C1;
Normal 0 1.
Print C1
Describe C1
Histogram C1
Random 500 C1;
Normal 0  3.
Random 500 C2;
Normal 0  1.
Random 500 C3;
Normal 0  1.
Name C1 "true"  C2 "x-error"  C3 "y-error"
Describe C1-C3
Add C1 C2 C4.
Add C1 C3 C5.
Name C4 "X" C5 "Y"
Info
Describe C4-C5
Count C4
Sum C4
Average C4
```

```
Medi  C4
Standard C4
Maxi C4
Mini C4
Histogram  C4
Histogram  C5
Plot C5 * C4;
symbol.
Correlation  C4 C5
Regress;
  Response 'Y';
  Continuous 'X';
  Terms 'X'.
Regress;
  Response 'Y';
  Continuous 'X';
  Terms 'X';
  Residuals C20;
  Coefficients C22.
Let C21=C5-C20
Plot C21 * C4;
symbol.
Plot C20 * C4;
symbol.
```

The Randomized Experimental Design

In this exercise you will simulate a simple pretest-posttest randomized experimental design. This design is of the form

R O X O
R O O

and thus has a pretest, a posttest, and two groups that have been randomly assigned. Note that in randomized designs a pretest is technically not required although one is often included as a covariate to increase the precision of program effect estimates. We will assume that we are comparing a program and comparison group (instead of two programs or different levels of the same program),

To begin, get into MINITAB in the usual manner. You should see the MTB prompt (which looks like this MTB>). Now you are ready to enter the following commands.

You will create two hypothetical tests as in previous exercises. Here, one test will be considered the pretest, the other the posttest. Assume that both tests measure the same true ability and that they each have their own unreliability or error:

```
MTB> Random 500 C1;
SUBC> Normal 50 5.
MTB> Random 500 C2;
SUBC> Normal 0 5.
MTB> Random 500 C3;
SUBC> Normal 0 5.
```

Here C1 represents the true ability on the tests for 500 people. C2 and C3 represent random error for the pretest and posttest respectively. Notice that the mean ability score for the tests will initially be set to 50 test score units. Next, construct the observed test scores:

```
MTB> Add C1 C2 C4.
MTB> Add C1 C3 C5.
```

You should notice that each test has about equal amounts of true score and error (because all three Random/Normal statements above use a 5 unit standard deviation). Now, name the columns:

```
MTB> Name C1 = 'true' C2 ='x error' C3 ='y error' C4 = 'pretest' C5 = 'posttest'
```

So far you have created a pretest and post for 500 hypothetical persons. Next, you need to randomly assign half of the people to the treated group and half to the control. One way to do this is to create a new random number for each individual. You will then use this variable to assign cases randomly. Since we want equal size groups (250 in each) you can assign all persons less than or equal to the median on this random number to one group, and all above the median to the other. Here is the way to do this:

```
MTB> random 500 C6;
SUBC> normal 0 5.
```

creates the random assignment number

```
MTB> let k1=min(C6)
MTB> let k2=median(C6)
MTB> let k3=max(C6)
```

gets the minimum, median and maximum values on this random assignment number. And

```
MTB> code (k1:k2) 0 (k2:k3) 1 c6 c7
```

creates the two equal size groups. To confirm that they are equal in size, do

```
MTB> table c7
```

and you should see that there are 250 0's and 1's.

Now, to be consistent with other exercises and to get rid of the unnecessary variable, put C7 into C6 and erase C7

```
MTB> let C6=C7 MTB> erase C7
```

Then, name C6

```
MTB> name C6='group'
```

Try the following three statements to verify that you have two groups of 250 persons:

```
MTB> Sign C6
MTB> Histogram 'Group'
```

Each of these presents slightly different information but both verify that you have two equal sized groups.

Now that you have created two groups, let's say that your treatment had an effect. To put in an effect you have to create a posttest score that has something added into it for those people who received the treatment, and does not add this in for the control cases. Remember that to create the posttest originally, you just added together the True Score and Posttest Error for each individual. To create the posttest with a 10-point treatment effect built in, you would use the following formula

$$Y = T + e_Y + (10* Z)$$

where Z is the 0,1 group variable (C6) you just created. To do this in MINITAB do

```
MTB> let c7=c1 + c3 + (10*c6)
MTB> name c7='postgain'
```

Now, c5 is the posttest when there is no treatment effect and c7 is the posttest when there is a 10-point treatment effect.

At this point, it's worth stopping and thinking about what you've done. You created a random True Score (C1) and added it to independent error (C2) to create a pretest (C4) and to other independent error (C3) to create a posttest (C5). Then you randomly assigned half of the people to a treatment (C6=1) and to a control (C6=0) condition. Finally, you created a posttest that has a 10-point treatment effect in it (C7). If

this were a real study (and not a simulation), you would observe only three variables: the pretest (X, C3), the group (Z, C6) and the posttest with a treatment effect in it (Y, C7).

Let's imagine how we might analyze the data using these three variables, in order to see whether the treatment has an effect. One of the first things we might do is to look at some simple distributions for the pretest and posttest. First, look at some histograms:

```
MTB> Histogram 'pretest'.

MTB> Histogram 'postgain'.

MTB> Histogram 'pretest';
SUBC> MidPoint;
SUBC> Bar 'group'.

MTB> Histogram 'postgain';
SUBC> MidPoint;
SUBC> Bar 'group'.
```

The first two commands show the histograms for all 500 cases while the last two show histograms for the two groups separately. Can you see that the two groups differ on average on the posttest?

Now, look at the bivariate distribution

```
MTB> Plot 'postgain' * 'pretest';
SUBC> Symbol 'group'.
```

You should see that the treated group has lots more high posttest scorers than the control group.

Now, look at some descriptive statistics tables.

```
MTB> Table 'Group';
SUBC> Means 'pretest' 'postgain';
SUBC> StDev 'pretest' 'postgain';
SUBC> N 'pretest' 'postgain'.
```

Here you should see clearly that while the two groups are very similar in average value on the pretest, they differ by nearly 10 points on the posttest.

In a randomized experiment, you technically don't need to measure a pretest. You could have the design:

R X O
R O

If you did, all you would be able to do to look for treatment effects is to compare the groups on the posttest. This might best be accomplished by conducting a simple t-test on the posttest

```
MTB> TwoT 95.0 c7 c6;
SUBC> alternative 0.
```

You can get the same result by using regression analysis with the following formula

$$Y = b_0 + b_1Z + e_Y$$

where

Y = posttest

Z = the 0,1 assignment variable

b_0 = posttest mean of the comparison group

b_1 = difference between the program and comparison group posttest means

e_Y = random error

This model can be run in MINITAB using

```
MTB> Regress 'postgain' 1 'Group'.
```

This regresses the posttest score onto the 0,1 group variable Z. The results for both the t-test and regression versions should be identical, but you have to know where to look to see this. In the t-test results, the last line will say in it 'T=' and report a t-value. The way you set up the simulation, this t-value should be negative in value (because it tests the control-treatment group difference which should be negative because the treatment group mean is larger by about ten points). Now look at the regression table under the heading 't-ratio'. The t-ratio for Group should be the same as the t-test result (except that the sign is reversed).

In general, the regression analysis method of testing for differences easier to use and interpret than the t-test results. In the regression results, b_0 is the coefficient for the Constant and b_1 is the coefficient for Group. The b_0 in this case is actually the average posttest value for the control group. The b_1 is the amount you add to the control group average to get the treatment group posttest average, that is, the estimate of the difference between the two groups on the posttest. This should be somewhere around 10 points. Both coefficients are tested with a t-test. The p-value tells you the probability that the estimated coefficient was obtained by chance.

So far, all you've done is to look at the difference between groups on the posttest. But you also have a pretest measured. How does this pretest help in analyzing the data? In a randomized experiment, the pretest (or any other covariate) is used to reduce variability in the posttest that is unrelated to the treatment. If you reduce posttest variability in this way, it should be easier to see a treatment effect. In other terms, for the very same posttest, including a good pretest should yield a higher t-value associated with the coefficient for differences between groups. To see this, you have to run a regression model that includes the pretest values in it. This model is:

$$Y = b_0 + b_1X + b_2Z + e_Y$$

where

Y = the posttest

X = the pretest

Z = the assignment variable

b_0 = the intercept of the comparison group line

b_1 = slope of regression lines

b_2 = the program effect

e_Y = random error

You can run this in MINITAB by doing:

MTB> Regress 'postgain' 2 'pretest' 'Group'.

Now, if you look at the t-ratio associated with the Group variable you should see that it is higher than it was in the original regression equation you ran. Even though you used the exact same posttest variable, you are able to see the treatment effect more clearly (i.e., got a higher t-value) because you included a good covariate (the pretest) that reduced some of the noise in the posttest that might obscure the treatment effect.

At this point you should be convinced of the following:

- One way to analyze data from this design is to conduct a t-test or one-way ANOVA on the difference between the posttest means. This can be accomplished using the simple model given earlier:

  $Y = b_0 + b_1Z + e_Y$

  Notice several things. First, this model fits regression lines for both groups, but because X is not included the lines have no slope (i.e., they are flat lines). You can construct the predicted line for both groups by substituting the appropriate values for Z. The regression line for the program group is:

  $Y_P = b_0 + b_1(1)$
  $Y_P = b_0 + b_1$

  and for the comparison groups it is:

  $Y_C = b_0 + b_1(0)$
  $Y_C = b_0$

  Therefore, the effect of the program is the difference between the two lines or

  $Y_P - Y_C = (b_0 - b_1) - b_0$
  $Y_P - Y_C = b_1$

  You should be convinced that this is the difference between the posttest means for the two groups.

- You also analyzed the data using a model called the analysis of covariance (ANCOVA):

  $Y = b_0 + b_1X + b_2Z + e_Y$

  This analysis is almost identical to the analysis used later for the regression-discontinuity design. Theoretically, one should get a similar estimate of the program effect with the ANCOVA and the

ANOVA but the ANCOVA estimate will in general be more precise. Specifically, the ANCOVA tests for posttest differences after "adjusting for" variance in the pretest. In general, then, given the same data and significance level it will be easier to find a significant effect (i.e., b_2 is not equivalent to 0) when using ANCOVA).

- The design simulated here is a very simple single-factor randomized experiment. You could simulate more complex designs. For example, to simulate a randomized block design you would first rank all persons on the pretest. Then you could set the block size, for example at n = 2. Then, beginning with the lowest two pretest scorers, you would randomly assign one to the program group and the other to the comparison group. You could do this by rolling a die--if you get a 1, 2, or 3 the lowest scorer is a program participant; if you get a 4, 5, or 6 the higher scorer is. Continuing in this manner for all twenty-five pairs would result in a block design. Designing a randomized block simulation of this type is difficult in MINITAB -- see if you can figure it out.

  You could also simulate a 2 x 2 factorial design. Here, you simply need to randomly assign four groups. You might want to assume that both programs are not equally effective and hence have different effect sizes for each. Also, in this case you would need to consider the interaction of the two factors and put in a specific effect size to simulate it. To develop such a factorial design in MINITAB you have to create two dummy-coded (0,1) variables to represent groups. You should also construct a variable representing their interaction (the easiest way is to multiply the two dummy-coded treatment variables together). You can then put in a treatment effect for either factor or for the interaction by adding some value associated with those terms into the posttest score. Your regression model will have to include the dummy-coded group variables and the interaction term.
- You might also change several of the key parameters of the simulation to see what happens. For instance, you might change the size of the error terms (C2 and C3) holding everything else constant, and look at what happens to the t-value associated with the treatment effect. Or, you might change the size of the treatment effect from 10 to 3 points and see how this affects the analysis.

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

Downloaded MINITAB Macro

```
GMACRO
RE
Random 500 C1;
Normal 50 5.
Random 500 C2;
Normal 0 5.
Random 500 C3;
Normal 0 5.
```

```
Add C1 C2 C4.
Add C1 C3 C5.
Name C1 = 'true' C2 ='x error' C3 ='y error' C4 = 'pretest' C5 = 'posttest'
random 500 C6;
normal 0 5.
let k1=min(C6)
let k2=median(C6)
let k3=max(C6)
code (k1:k2) 0 (k2:k3) 1 c6 c7
table c7
let C6=C7
erase C7
name C6='group'
Sign C6
Histogram 'Group'
let c7=c1 + c3 + (10*c6)
name c7='postgain'
Histogram 'pretest'.
Histogram 'postgain'.
Histogram 'pretest';
MidPoint;
Bar 'group'.
Histogram 'postgain';
MidPoint;
Bar 'group'.
Plot 'postgain'  * 'pretest';
Symbol 'group'.
Table 'Group';
Means 'pretest' 'postgain';
StDev 'pretest' 'postgain';
N 'pretest' 'postgain'.
TwoT 95.0 c7 c6;
alternative 0.
Regress 'postgain' 1 'Group'.
Regress 'postgain' 2 'pretest' 'Group'.
ENDMACRO
```

The Nonequivalent Group Design - Part I

In this exercise you are going to create data for and analyze a nonequivalent group design. The design has several important characteristics. First, a pretest and posttest are given to all participants. Second, the design usually has two groups, one which gets some program or treatment and one which does not (usually termed the "program" and " comparison groups respectively). Third, the two groups are "nonequivalent groups", that is, we expect that they may differ prior to the study. Often, nonequivalent groups are simply two intact groups that are accessible to the researcher (e.g., two classrooms, two states, two cities, two mental health centers, etc.). We can depict the design using the following notation:

N O X O
N O O

where the N indicates that the groups are nonequivalent, the first O represents the pretest, the X indicates administration of some program or treatment, and the last O signifies the posttest. Notice that the top line represents the program group while the bottom line signifies the comparison group.

To begin, get into MINITAB in the usual manner. You should see the MTB prompt (which looks like this MTB>). Now you are ready to enter the following commands.

You will create two hypothetical tests as in previous exercises. Here, one test will be considered the pretest, the other the posttest. We will assume that both tests measure the same true ability and that they each have their own unreliability or error:

```
MTB> Random 500 C1;
SUBC> Normal 50 5.
MTB> Random 500 C2;
SUBC> Normal 0 5.
MTB> Random 500 C3;
SUBC> Normal 0 5.
```

Here C1 represents the true ability on the tests for 500 people. C2 and C3 represent random error for the pretest and posttest respectively. Notice that the mean ability score for the tests will initially be set to 50 test score units. Next, construct the observed tests scores:

```
MTB> Add C1 C2 C4.
MTB> Add C1 C3 C5.
```

You should notice that each test has about equal amounts of true score and error (because all three Random/Normal statements above use a 5 unit standard deviation). Now, name the columns:

```
MTB> Name C1 ='true' C2 ='x error' C3 ='y error' C4 ='pretest' C5 ='posttest'
```

What you have done so far is to create a pretest and post for 500 hypothetical persons. Next, you have to create "nonequivalent" groups. For convenience, you will create groups of 250 persons each. To do this enter:

```
MTB> Set C6
DATA> 1:500
DATA> End
MTB> Code (1:250) 0 C6 C6
MTB> Code (251:500) 1 C6 C6
```

The SET statement (and the two associated DATA statements) simply numbers each person from 1 to 500 and puts this sequence of numbers in C6. The first code statement essentially says "change all the numbers from 1 to 250 in C6 to 0's and put these 0's back into C6." The second code replaces the numbers from 251 to 500 with a 1. You have created two groups of 250 persons each. You know which group a person is in by looking at their value in C6. If they have a 0, they are in one group; if they have a 1, they are in the other. For convenience, the persons having a zero will be the comparison group and those having a one will be the program group. You should name this new variable:

```
MTB> Name C6 = 'Group'
```

Try the following three statements to verify that you have two groups of 250 persons:

```
MTB> Table C6
MTB> Sign C6
MTB> Histogram 'Group'
```

Each of these presents slightly different information but all of them verify that you have two equal sized groups.

But you have still not created "nonequivalent" groups. To see this, you will use the subcommand form of the TABLE command:

```
MTB> Table C6;
SUBC> means C4 C5.
```

The first row of the table gives the pretest and posttest means for the comparison group (C6 = 0) while the second row gives these values for the program group. At this point, all four means should be near 50 test score units.

In the nonequivalent group design we typically select two groups which we hope are similar or equivalent. Nevertheless, because we don't select these groups randomly, we expect that one group may be better or worse than the other before our study. You saw from the table command above that both groups appear to be similar on the pretest. Therefore, you can create nonequivalent groups by making the program group slightly better in test ability. This situation might occur in real life if we chose two classrooms of students that we thought were pretty similar, only to find out that one group scores on the average a few points better than the other. To create the "advantaged" program group do the following:

```
MTB> Let C4 = C4 + (5 * C6)
MTB> Let C5 = C5 + (5 * C6)
```

It is important to think about what these statements are doing. The first let command operates on the pretest scores. You add five test score points to each program group pretest score. How does this work? Remember that C6 has a 0 for all the comparison group persons and a 1 for the program people. When you multiply 5 times this C6 variable the result will be a zero for each comparison person and a 5 for

each program person. You then add these 0 or 5 points to the original pretest score and put the result right back into C4. The second Let command does the same thing for the posttest scores and, as a result, this "advantage" should be seen on both the pre and posttest. Now verify that you have an "advantaged" program group (that is, that you have nonequivalent groups). You will again use the table command but will add another subcommand to give the standard deviations:

```
MTB> Table C6;
SUBC> means C4 C5;
SUBC> stdev C4 C5.
```

Clearly, the program group has pre and posttest averages in the vicinity of 55 test score units.

So far, you have created two nonequivalent groups having a pretest and posttest. But one of these groups received your program or treatment. Did it work? It would appear from the data that it did not. The difference between the group means on the pretest is about the same as their posttest difference. About the only way that you could claim that the program had an effect is if you had reason to believe that without it the posttest difference between the means would have been different than it is. This would be possible, for example, if the groups had been maturing at different rates (a selection - maturation threat) but without any other evidence than these test scores this would be a hard argument to accept. On the basis of this data you would probably conclude that the program was ineffective. This makes sense especially because you did not build into the data any program effect. Now add 10 test score points to the posttest for each program person - a treatment effect of 10 points:

```
MTB> Let C5 = C5 + (10*C6)
```

which you should recognize as the same type of command that you used above to create nonequivalent groups in the first place. Now, look at the means and standard deviations:

```
MTB> Table C6;
SUBC> means C4 C5;
SUBC> stdev C4 C5.
```

Now, the pretest difference between the two groups is still about 5 points on the average, but the posttest difference is about 15 points. The "gain" of the program group over what you might expect on the basis of pretest scores appears to be about 10 points (which, of course, is exactly what you set it up to be).

At this point it is worth reflecting on what you have done. If you had conducted a study using the nonequivalent group design, you could have obtained data like that which is described in the last table. You would notice that the groups appear to be different on the pretest, with the program group having the advantage. You would also notice that the difference between the groups is considerably larger on the posttest. In fact, you have simulated data for a nonequivalent group design and (whether you realize it or not) you have explicitly controlled the size of the correlation between the measures, the number of persons in each group, the amount of nonequivalence between the groups, the size of the program effect, and so on. One reason we run simulations of this type is to determine whether statistical analyses which we use give us accurate estimates of the effect of the effect of our programs. Since we specifically put in a 10 point program effect, we would expect that an accurate analysis would tell us that the effect was about that large. Let's find out if our analysis will work.

The typical strategy for analyzing pretest-posttest group designs is one which is based on the Analysis of Covariance (ANCOVA). Essentially, we want to look at the difference between the two groups on the posttest after we have "adjusted for" the initial differences between the groups as measured on our covariate - the pretest. The ANCOVA can be analyzed using multiple regression analysis (you should recognize that the ANCOVA is simply a subset of the multiple regression model - we would get exactly the same results for the analysis whether we use a computer program which does ANCOVA or one which does regression as long as we tell the regression program the correct model to estimate. We will generally use the regression command in MINITAB to conduct an Analysis of Covariance).

Before actually running the analysis you ought to look at plots of the data. Try some of the following:

```
MTB> Histogram C4
MTB> Histogram C5
MTB> Plot C5 * C4
MTB> Plot C5 * C4;
SUBC> symbol C6.
```

The Histogram commands show the distributions for the pretest and posttest. The first plot command shows the pre-post bivariate distribution. The second plot command shows this same distribution but uses different symbols for the program and comparison groups. Unfortunately, it may be difficult to see the program and comparison groups distinctly. As a side exercise, you might try to use the choose command to create separate columns of pre and posttest scores for the two groups so these distributions can be plotted separately. Now run the ANCOVA using the MINITAB regression command. The regression model form of the ANCOVA can be stated as:

$$Y = b_0 + b_1X + b_2Z + e_Y$$

where

Y = the posttest (C5)

X = the pretest (C4)

Z = the assignment variable (C6)

b_0 = the intercept of the comparison group line

b_1 = slope of regression lines

b_2 = the program effect

e_Y = random error

To do this analysis enter:

```
MTB> Regress C5 2 C4 C6
```

The computer will first print out the regression equation. The first number on the right of the equal sign is the intercept (b_0)of the comparison group regression line (because you included C6, a dummy 0,1 variable in the regression, in effect two lines are being fit to the data, one for each group). The second number in the equation gives the slope (b_1) for the program and comparison group regression lines (recall that the Analysis of Covariance assumes that the slopes of the two groups are equal - thus, we

only simulate a single value). The third number after the equal sign is the estimate of the program effect (b_2). Recall that you put in a program effect of 10 points. Is this value close? The table below the equation tests whether these three values are significantly different from zero. Since you are particularly interested in determining whether this analysis gives an accurate estimate of the program effect, you should look in the table for the line for variable C6, the "group " variable. The coefficient or estimate that was shown in the equation is repeated first on this line. Then the standard deviation of the estimate is shown. You know that you put in a program effect of 10 points. To see whether the estimate given by the analysis is accurate at a .05 level of significance, you have to construct a confidence interval for the estimate or coefficient. To do this, first multiply the standard deviation for that coefficient by 2 and then add and subtract this value from the estimate. For example, let's say the analysis tells you that the estimate or coefficient of the C6 variable is 11.3 and that the standard deviation is 0.5 units. Given this, the .05 confidence interval ranges from 10.3 to 12.3 (that is 11.3 plus or minus 2 times .5) . This analysis would be telling you that the best estimate of the program effect is 11.3 and that the odds are less than 5 out of 100 that the true effect is outside of that range. Recall that you have simulated that the true effect is ten points. In this example, you would wonder whether the analysis we used (ANCOVA) is working correctly because the program effect that you put in doesn't fall within the 95% confidence interval. When you construct the confidence interval do you find that 10 is included within it or not? Is the estimate of effect above or below 10?

If you have followed the instructions, you will find that most of the time you will <u>not</u> get an accurate estimate of the effect. In fact, ANCOVA yields biased estimates of effect for this type of nonequivalent group design. We do have better analysis strategies, but in order to understand them well it is important to understand why the ANCOVA strategy fails. You should try to get some idea of why ANCOVA fails by conducting simulations like the one above. Some variations are suggested below. The next exercise will present an analysis strategy which can often be used to obtain correct estimates of program effect.

- A key reason for the failure of ANCOVA is unreliability or error in the measures. You explicitly controlled the reliability by setting the standard deviations of the true and error scores in the Random/Normal statements. Try the simulation again setting the true score standard deviation to 10 and the error standard deviations to 1.
- Try the variation above but make the pretest more reliable than the posttest. To do this, use a small standard deviation for the pretest error (C2) and a larger one for posttest error (C3).
- Try to construct a simulation where the treatment group is disadvantaged relative to the comparison group. To do this, you will have to multiply the C6 variable by a negative number in the appropriate let statement above.
- Put in a negative program effect. To do this you will have to use a negative number where you used the +10 above. A negative effect implies that your program actually hurt rather than helped the program group relative to the comparison group.

When we use simulation techniques to investigate the accuracy of a statistical analysis we never rely on the results of a single run because the results could be wrong simply by chance. Typically, we would run the simulation several hundred times and average the estimates of program effect to see if the analysis is biased or not. Although that many runs is probably not feasible for you, it might be worthwhile for you to compare the estimates of effect that you got with estimates which others obtain. If you average these estimates, you should see more clearly that ANCOVA yields a biased estimate for the nonequivalent group design.

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

```
GMACRO
NEGD1
Random 500 C1;
Normal 50 5.
Random 500 C2;
Normal 0 5.
Random 500 C3;
Normal 0 5.
Add C1 C2 C4.
Add C1 C3 C5.
Name C1 ='true' C2 ='x error' C3 ='y error' C4 ='pretest' C5 ='posttest'
Set C6
1:500
End
Code (1:250) 0 C6 C6
Code (251:500) 1 C6 C6
Name C6 = 'Group'
Table C6
Sign C6
Histogram 'Group'
Table C6;
means C4 C5.
Let C4 = C4 + (5 * C6)
Let C5 = C5 + (5 * C6)
Table C6;
means C4 C5;
stdev  C4 C5.
Let C5 = C5 + (10*C6)
Table C6;
means C4 C5;
stdev  C4 C5.
Histogram C4
Histogram C5
Plot C5 * C4;
symbol.
Plot C5  * C4;
Symbol C6.
Regress C5 2 C4 C6
ENDMACRO
```

The Nonequivalent Group Design - Part II

This is the second exercise in the analysis of nonequivalent group designs. In the first exercise you learned how to simulate data for a pretest - posttest two-group design. You put in an initial difference between the two groups (i.e., a selection threat) and also put in a treatment effect. You then analyzed the data with an Analysis of Covariance (ANCOVA) model and, if all went well, you found that the estimate of program effect was biased (i.e., the true effect was not within a 95% confidence interval of the obtained estimate).

In this exercise, you will create data exactly as you did in the last exercise. This time however, you will adjust the data for unreliability on the pretest before conducting the ANCOVA. Because unreliability or measurement error on the pretest causes the bias when using ANCOVA, this correction should result in an unbiased estimate of the program effect.

By now you should be fairly familiar with what most of these MINITAB commands are doing (remember, you should consult the MINITAB Handbook or use the HELP command if you want information about a command) and so the initial command will be presented without explanation. Get into MINITAB and you should see the MINITAB prompt (which looks like this MTB>). Now you are ready to enter the following commands:

```
MTB> Random 500 C1;
SUBC> Normal 50 5.
MTB> Random 500 C2;
SUBC> Normal 0 5.
MTB> Random 500 C3;
SUBC> Normal 0 5.
MTB> Add C1 C2 C4.
MTB> Add C1 C3 C5.
MTB> Name C1 ='true' C2 ='x error' C3 ='y error' C4 ='pretest' C5 ='posttest'
MTB> Set C6
DATA> 1:500
DATA> End
MTB> Code (1:250) 0 C6 C6
MTB> Code (251:500) 1 C6 C6
MTB> Name C6 = 'Group'
MTB> Let C4 = C4 + (5 * C6)
MTB> Let C5 = C5 + (15 * C6)
```

You have constructed the data exactly as in Part I. Notice that the data commands from the previous exercise have been condensed here. If you don't understand why you used these commands, go back and read the instructions for Part I again. Be sure that you understand that you now have data for two groups, one of which received a program or treatment (i.e., those with a '1' for C6) and the other a comparison group (i.e., with a '0' for C6). The groups differ on the pretest by an average of five points, with the program group being advantaged. On the posttest, the groups differ by fifteen points on the average. It is assumed that five of this fifteen point difference is due to the initial difference between groups and that the program had a ten point average effect. You should verify this by examining the group means:

```
MTB> Table C6;
SUBC> means C4 C5.
```

You should also look at histograms and at the bivariate distribution:

```
MTB> Histogram C4
MTB> Histogram C5
MTB> Plot C5 * C4
MTB> Plot C5 * C4;
SUBC> Symbol C6.
```

Recall that the ANCOVA yields biased estimates of effect because there is measurement error on the pretest. You added in the measurement error when you added C2 (x-error) into the pretest score. In fact, you added in about as much error as true score because the standard deviations used in all three Random/Normal commands are the same (and equal to 5). In order to adjust the pretest scores for measurement error or unreliability, we need to have an estimate of how much variance or deviation in the pretest is due to error (we know that it is about half because we set it up that way). Recall that reliability is defined as the ratio of true score variance to total score variance and that we estimate reliability using a correlation. Traditionally, we can use either a split-half reliability (the correlation between two randomly-selected subsets of the same test) or a test-retest correlation. In these simulations we will correct the pretest scores using a test-retest correlation. Since the test-retest correlation may differ between groups it is often advisable to calculate this correlation for the program and comparison groups separately. Before we can do this we have to separate the data for the two groups and put it in separate columns:

```
MTB> Copy C6 C4 C5 C20-C22;
SUBC> use C6 = 0.
MTB> Copy C6 C4 C5 C30-C32;
SUBC> use C6 = 1.
MTB> Name C21 = 'contpre' C22 = 'contpost' C31= 'progpre' C32 = 'progpost'
```

The first statement copies all cases having a '0' in C6 (i.e., all the comparison group cases) and their pretest and posttest scores (C4 and C5) and puts these scores in C20 through C22. It is important for you to notice that the order in which you copy the variables (i.e., C6, C4, C5) is the order in which they are put into the other columns. Therefore, C20 should consist entirely of '0' values. The next statement copies all the program group cases and puts them in C30 through C32. We are not interested in C20 or C30 because these consist of the '0' and '1' values so we don't name them. We name C21 'contpre' to stand for control pretest, C31 'progpre' to stand for program group pretest, and so on. Now, we are ready to estimate the test- retest reliabilities for each group. These are simply the correlations between the pretest and posttest for each group:

```
MTB> Correlation C21 C22 M1
MTB> Correlation C31 C32 M2
```

Each correlation is stored in a 2x2 matrix variable. In order for us to use these correlations later, we have to copy them from the M matrix into a K-variable constant. We can do this with the following commands:

```
MTB> copy M1 C41 C42
MTB> copy C41 K1;
```

```
SUBC> use 2.
MTB> copy M2 C43 C44
MTB> copy C43 K2;
SUBC> use 2.
```

Now, we will adjust the pretest scores for the comparison group. All you need to do is the following formula:

```
MTB> Let C23 = aver (C21) + (K1 * (C21 - aver (C21)))
```

Be sure to type this statement exactly as is. If you make a mistake, just re-enter the command. Now, let's adjust the scores for the program group. Use the following formula:

```
MTB> Let C33 = aver (C31) + (K2 * (C31 - aver (C31)))
```

You should recognize that for each group you are taking each person's pretest score, subtracting out the group pretest average, multiplying this difference by the within-group pre-post correlation (the K-variable), and then adding the group pretest mean back in. For instance, he formula for this reliability correction is (note that X_{mean} is the mean within each group):

$$X_{adj} = X + r_{xx}(X_i - X_{mean})$$

where:

X_{adj} = the adjusted or reliability corrected pretest

X_{mean} = the within-group pretest mean

X_i = pretest score for case i

r_{xx} = an estimate of pretest reliability

We should probably name these new adjusted pretests:

```
MTB> Name C23='contadj' C33='progadj'
```

Now, compare the pretest means for each group before and after making the adjustment:

```
MTB> Describe C21 C22 C23 C31 C32 C33
```

What do you notice about the difference between unadjusted and adjusted scores? Did the mean change because of the adjustment (compare the means for C21 and C23 for instance)? Did the standard deviations? Remember that about half of the standard deviation of the original scores was due to error. How much have we reduced the standard deviations by adjusting? How big is the pre-post correlation for each group? Is the size of the correlation related to the standard deviations in any way?

Now, we want to combine the pretest scores for the two groups back into one set of scores (i.e., one column of data). To do this we use the Stack command:

```
MTB> Stack (C23) (C33) (C7).
```

Which means 'stack the scores in C23 on top of the scores in C33 and put these into C7'. Now let's name the adjusted pretest scores:

```
MTB> Name C7 = 'adjpre'
```

Now take a look at the means and standard deviations of the original pretest, adjusted pretest and posttest by group:

```
MTB> Table C6;
SUBC> means C4 C7 C5;
SUBC> stdev C4 C7 C5.
```

It should be clear to you that the adjustment for measurement error on the pretest did not affect the means, but did affect the standard deviations. You are now ready to conduct the analysis. First, do the ANCOVA on the original pretest scores (C4) just like in Part I. You should see that the estimate of effect is biased. Then, conduct the analysis using the adjusted pretest scores (C7). This time you should see that the estimate of effect is much closer to the ten points that you originally put in. First, the biased analysis:

```
MTB> Regress C5 2 C4 C6
```

Remember that the estimate of the program effect is the COEFFICIENT in the table which is on the C6 GROUP line. Is it near a value of ten? You can construct a 95% confidence interval for this estimate using the ST.DEV. OF COEF. which is on the same line as the estimate. To construct this interval, first multiply the standard deviation of the coefficient by two (remember from statistics that the 95% confidence interval is plus or minus approximately two standard error units). To get the lower limit of the interval subtract this value from the coefficient in the table, to get the upper limit add them. Does the true program effect (10 points) fall within the bounds of this interval. For most of you it will not (just by chance, it may for a few of you. Event if it does fall within the interval it will probably not be near the center of the interval). Now, conduct the same ANCOVA analysis this time substituting the adjusted pretest scores (C7) for the original ones:

```
MTB> Regress C5 2 C7 C6
```

Again, look at the coefficient in the table that is on the same line as the C6 GROUP variable. Is this value closer to the true program effect of 10 points than it was for the previous analysis? It should be (again, for some of you this may not be true just by chance alone). Once again, construct the 95% confidence interval for the coefficient using the standard deviation of the coefficient given in the table. You should see that this time the true program effect of 10 points falls within the interval.

You have just conducted a reliability-corrected Analysis of Covariance. In practice, we know that test-retest and split-half reliability estimates will often differ, sometimes considerably. Because of this, we will often conduct two analyses like the one above- once using the test-retest correlations and once using the split-half correlations. Although this will give us two estimates of the program effect, we expect that one is a conservative adjustment while the other is a liberal one. We can therefore be fairly confident that the true program effect probably lies somewhere between the two estimated effects.

At this point, you should stop and consider the steps that were involved in conducting this analysis. You might want to try one or more of the following variations:

- Change the reliability of the pretest. Recall that the reliability of the pretest depends on how much true score and error you add into it. Try the simulation with extremely low reliability (high standard deviation for error, C2, low for true score, C1) and high reliability. You might even want to try a perfectly reliable pretest (i.e., don't add C2 in at all). In this case, we would expect that the unadjusted ANCOVA yields an unbiased estimate of the program effect.
- Change the reliability of the posttest. Here you would alter the ratio of the standard deviations for the Y error, C3, and true score, C1 variables. Recall that unreliability on the posttest should not result in biased estimates with unadjusted ANCOVA. However, the estimates will be less precise when you have more error on the posttest. You will see this by looking at the standard deviation of the coefficient and comparing it with the standard deviation of the coefficient which you obtained in this simulation. With higher error variance on the posttest, the 95% confidence interval should be wider.
- As in the previous exercise variations, try to construct a simulation where the program group is disadvantaged relative to the comparison group.
- Conduct the simulation above but put in a negative program effect. To do this, just put in a -15 for the 15 in the LET statement which constructed the posttest.

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

```
GMACRO
NEGD2
Random 500 C1;
Normal 50 5.
Random 500 C2;
Normal 0 5.
Random 500 C3;
Normal 0 5.
Add C1 C2 C4.
Add C1 C3 C5.
Name C1 ='true' C2 ='x error' C3 ='y error' C4 ='pretest' C5 ='posttest'
Set C6
1:500
End
Code (1:250) 0 C6 C6
Code (251:500) 1 C6 C6
Name C6 = 'Group'
Let C4 = C4 + (5 * C6)
Let C5 = C5 + (15 * C6)
Table C6;
```

```
means C4 C5.
Histogram C4
Histogram C5
Plot C5 * C4
Plot C5 * C4;
Symbol C6.
Copy C6 C4  C5 C20-C22;
use C6 = 0.
Copy C6 C4 C5 C30-C32;
use C6 = 1.
Name C21 = 'contpre' C22 = 'contpost' C31='progpre' C32 = 'progpost'
Correlation C21 C22 M1
Correlation C31 C32 M2
copy M1 C41 C42
copy C41 K1;
use 2.
copy M2 C43 C44
copy C43 K2;
use 2.
Let C23 = aver (C21) + (K1 * (C21 - aver(C21)))
Let C33 = aver (C31) + (K2 * (C31 - aver(C31)))
Name C23='contadj' C33='progadj'
Describe C21 C22 C23 C31 C32 C33
Stack (C23) (C33) (C7).
Name C7 = 'adjpre'
Table C6;
means C4 C7 C5;
stdev C4 C7 C5.
Regress C5 2 C4 C6
Regress C5 2 C7 C6
ENDMACRO
```

The Regression Discontinuity Design

In this exercise we are going to create and analyze data for a regression discontinuity design. Recall that in its simplest form the design has a pretest, a posttest, and two groups, usually a program and comparison group. The distinguishing feature of the design is its procedure for assignment to groups -- persons or units are assigned to one or the other group solely on the basis of a cutoff score on the pre-program measure. Thus, all persons having a pre-program score on one side of the cutoff value are put into one group and all remaining persons are put in the other. We can depict the design using the following notation:

C O X O
C O O

where the C indicates that groups are assigned by a cutoff score, the first O represents the pretest, the X depicts the administration of some program or treatment and the second O signifies the posttest. Notice that the top line represents the program group while the second line indicates the comparison group.

In this simulation you will create data for a "compensatory" program case. We assume that both the pretest and posttest are fallible measures of ability where higher scores indicate generally higher ability. We also assume that we want the program being studied to be given to the low pretest scorers - those whose are low in pretest ability.

Get into MINITAB as you normally would. You should see the MINITAB prompt . Now you are ready to enter the commands below.

The first step is to create two hypothetical tests, the pretest and posttest. Before you can do this you need to create a measure of true ability and separate error measures for each test:

```
MTB> Random 500 C1;
SUBC> Normal 50 5.0.
MTB> Random 500 C2;
SUBC> Normal 0 5.0.
MTB> Random 500 C3;
SUBC> Normal 0 5.0.
```

Now you can construct the pretest by adding true ability (C1) to pretest error (C2):

```
MTB> Add C1 C2 C4
```

Before constructing the posttest it is useful to create the variable that describes the two groups. The pretest mean will be about 50 and you will use 50 as the cutoff score in this simulation. Because this is a compensatory case, we want all those who score lower than or equal to 50 to be program cases, with all those scoring above 50 to be in the comparison group. The following two code statements will create a new dummy variable (C5) with a value of 1 for program cases and 0 for comparison cases:

```
MTB> Code (0:50) 1 C4 C5
MTB> Code (50:100) 0 C5 C5
```

To check on how many persons you have in each condition do:

```
MTB> Table C5
```

Notice that you probably don't have exactly 250 people in each group (although in the long run, that is how many you would expect if you divide a normal distribution at the mean). Now you are ready to construct the posttest. We would like to simulate an effective program so we will add in 10 points for all program cases (recall that you accomplish this by multiplying 10 by the dummy-coded treatment variable - for all program cases this product is 10, for comparison cases, 0 -- this is then added into the posttest):

```
MTB> Let C6= C1 + C3 + (10*C5)
```

It is convenient to name the variables:

```
MTB> Name C1 = 'true' C2 ='x error' C3 ='y error' C4 ='pretest' C5='group' C6='posttest'
```

To get some idea of what the data look like try:

```
MTB> Table C5;
SUBC> means C4 C6.
```

and don't forget to put the period at the end of the second line. This command gives pre and post means for the two groups. Note that the program group starts off at a distinct disadvantage - we deliberately selected the lower scorers on the pre-program measure. Notice also that the comparison group actually regresses back toward the overall mean of 50 between the pretest and posttest. This is to be expected because you selected both groups from the extremes of the pretest distribution. Finally, notice that the program group scores as well or better than the comparison group on the posttest. This is because of the sizable 10 point program effect which you put in. You might examine pre and post histograms, correlations, and the like. Now, look at the bivariate distribution:

```
MTB> Plot C6 * C4;
SUBC> symbol C5.
```

You should be able to see that the bivariate distribution looks like it "jumps" at the pretest value of 50 points. This is the discontinuity that we expect in a regression-discontinuity design when the program has an effect (note that if the program has no effect we expect a bivariate distribution that is continuous or does not jump).

At this point you have finished creating the data. The distribution that you see might be what you would get if you conducted a real study (although real data seldom behaves as well as this). The first step in analyzing this data is to examine the data to try to determine what the "likely" pre-post function is. We know that the true function here is linear (that the same straight-line fit in both groups is appropriate) but with real data it will often be difficult to tell by visual inspection alone whether straight or curved lines are needed. Therefore even though we might think that the most likely function or distribution is linear, w e will deliberately over-fit or over-specify this likely function a bit to be on the safe side.

The first thing you need to do to set up the analysis is to set up a new variable that will assure the program effect will be estimated at the cutoff point. To do this, you simply create a new variable that is equal to the pretest *minus the cutoff score*. You should see that this new variable will now be equal to zero at the cutoff score and that all program cases will have negative values on this score while the

comparison group will have positive ones. Since the regression program would automatically estimate the vertical difference or "jump" between the two groups at the intercept (i.e., where the pretest equals 0), when you create this new variable you are setting the cutoff equal to the a pretest value of 0 and the regression program will correctly estimate the jump at the cutoff. Put this new variable in C7:

```
MTB> Let C7 = C4 - 50
MTB> Name C7 = 'pre-cut'
```

and name it appropriately. You will see that we always substitute this variable for the pretest in the analyses.

Now you need to set up some additional variables that will enable you to over-specify the "likely" true linear function:

```
MTB> Let C8 = C7 * C5
```

This new variable is simply the product of the corrected pretest and the dummy assignment variable. Thus C8 will be equal to zero for each comparison group case and equal to the corrected pretest for each program case. When this variable is added into the analysis we are in effect telling the regression program to see if there is any interaction between the pretest (C7) and the program (C5). This is equivalent to asking whether the linear slopes in the two groups are equal or whether they are different (which implies that the effect of the program differs depending on what pretest score a person had). Now, construct quadratic (second-order) terms:

```
MTB> Let C9 = C7 * C7
MTB> Let C10 = C9 * C5
```

For C9 you simply square the pretest. When this variable is entered into the analysis we are in effect asking whether the bivariate distribution looks curved in a quadratic pattern (consult an introductory algebra book if you don't recall what a quadratic or squared function looks like). The second variable, C10, allows the quadratic elements in each group to differ, and therefore, can be considered a quadratic interaction term. You should name the variables:

```
MTB> Name C8 ='I1' C9 = 'pre2' C10 = 'I2'
```

where I1 stands for 'linear interaction' , PRE2 for the 'squared pretest', and I2 for the 'quadratic interaction'. You could continue generating even higher-order terms and their interactions (cubic, quadratic, quintic etc.) but these will suffice for this demonstration.

You are now ready to begin the analysis. You will do this in a series of regression steps, each time adding in higher-order terms. Because of the length of the standard regression output, you might want to request briefer output with the command

```
MTB> Brief 1
```

In the first step, you fit a model which assumes that the bivariate distribution is best described by straight lines with the same slopes in each group and a jump at the cutoff:

```
MTB> Regress C6 2 C7 C5
```

This is simply the standard Analysis of Covariance (ANCOVA) model. The coefficient associated with the GROUP variable in the table is the estimate of the program effect. Since you created the data you know that this regression analysis exactly-specifies the true bivariate function - you created the data to have the same slope in each group and to have a program effect of 10 points. Is the estimate that you obtained near the true effect of ten? You can construct a 95% confidence interval (using plus or minus 2 times the standard deviation of the coefficient for the GROUP variable). Does the true effect of ten points fall within this interval (it should for almost all of you)?

With real data we would not be sure that the model we fit in this first step includes all the necessary terms. If we have left out a necessary term (for instance, if there was in fact a linear interaction) then it would be very likely that the estimate we obtained in this first analysis would be biased (you will see this in the next simulation). To be on the safe side, we will add in a few more terms to the analysis in successive regression steps. If we have already included all necessary terms (as in the analysis above) then these additional terms should be superfluous. They should not bias the estimate of program effect, but there will be less precision. For the next step in the analysis you will allow the slopes in the two groups to differ by adding in Il, the linear interaction term:

```
MTB> Regress C6 3 C7 C5 C8
```

The coefficient for the GROUP variable is, as usual, the estimate of program effect. We know that the new variable, C8, is unnecessary because you set up the simulation so that the slopes in both groups are the same. You should see that the coefficient for this Il variable is near zero and that a zero value almost surely falls within the 95% confidence interval of this coefficient. Because this term is unnecessary, you should still have an unbiased estimate of the program effect. Is the coefficient for GROUP near the true value of 10 points? Does the value of 10 fall within the 95% confidence interval of the coefficient? You should also note that the estimate of the program effect is less precise in this analysis than in the previous one - the standard error of the coefficient for the GROUP variable should be larger in this case than in the previous run. Now, add in the quadratic term:

```
MTB> Regress C6 4 C7 C5 C8 C9
```

Again, you should see that the coefficients for the superfluous terms (I1 and PRE2) are near zero. Similarly, the estimate of program effect should still be unbiased and near a value of ten. This time the standard error of the GROUP coefficient will be a little larger than last time - again indicating that there is some loss of precision as higher-order terms are added in. Finally, you will allow the quadratic terms to differ between groups by adding in the quadratic interaction term, I2:

```
MTB> Regress C6 5 C7 C5 C8 C9 C10
```

By now you should be able to see the pattern across analyses. Unnecessary terms will have coefficients near zero. The program effect estimate should still be near ten, but the 95% confidence interval will be slightly wider indicating that there is a loss of precision as you add in more terms.

In an analysis of real data, you would by now be more convinced that your initial guess that the bivariate distribution was linear was a sensible one. You might decide to continue fitting higher order terms or you might stop with the quadratic terms. This whole procedure may strike you as somewhat wasteful. If we think the correct function is linear, why not just fit that? The procedure outlined here is a conservative one. It is designed to minimize the chances of obtaining a biased estimate of program effect by increasing your chances of overspecifying the true function. At this point you should stop and

consider the steps that were involved in conducting this analysis. You might want to try one or more of the following variations:

- Change the reliability of the pretest. Recall that the reliability of the pretest depends on how much true score and error you add into it. Try the simulation with extremely low reliability (high standard deviation for error, C2, low for true score, C1) and high reliability. You might even want to try a perfectly reliable pretest (i.e., don't add C2 in at all). What effect does pretest reliability have on estimates of the treatment effect? How is this different from or similar to the nonequivalent group design?
- Change the reliability of the posttest. Here you would alter the ratio of the standard deviations for the Y error, C3, and true score, C1 variables. How does posttest reliability relate to the estimate of the effect?
- Construct a simulation where the program group consists of the high pretest scorers. In what kinds of real-world situations would the high scorers be likely to receive a new program?
- Put in a negative program effect. To do this, just put in a -10 for the 10 in the LET statement which constructed the posttest. How does the shape of the bivariate distribution change when you introduce a negative rather than a positive effect?

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

```
GMACRO
RD
Random 500 C1;
Normal 50 5.0.
Random 500 C2;
Normal  0  5.0.
Random 500 C3;
Normal  0  5.0.
Add C1 C2 C4
Code (0:50) 1 C4 C5
Code (50:100) 0 C5 C5
Table C5
Let C6= C1 + C3 + (10*C5)
Name  C1 = 'true'  C2 ='x error'   C3 ='y error'  C4 ='pretest'   C5='group'  C6='posttest'
Table C5;
means C4 C6.
Plot C6  * C4;
Symbol C5.
Let C7 = C4 - 50
```

```
Name C7 = 'pre-cut'
Let C8 = C7 * C5
Let C9 = C7 * C7
Let C10 = C9 * C5
Name C8 ='I1'  C9 = 'pre2' C10 = 'I2'
Brief 1
Regress C6 2 C7 C5
Regress C6 3 C7 C5 C8
Regress C6 4 C7 C5 C8 C9
Regress C6 5 C7 C5 C8 C9 C10
ENDMACRO
```

Regression Artifacts

In this exercise you are going to look at how regression to the mean operates. It is designed to convince you that regression effects do occur, when they can be expected to occur, and to hint at why they occur.

To begin, get into MINITAB as usual. You should see the MINITAB prompt (which looks like this MTB>). Now you are ready to enter the following commands. You will begin by creating two variables similar to the ones in the Generating Data exercise:

```
MTB> Random 500 C1;
SUBC> Normal 50 10.
MTB> Random 500 C2;
SUBC> Normal 0 5.
MTB> Random 500 C3;
SUBC> Normal 0 5.
```

You have created three random variables. The first will have a mean or average of 50 and a standard deviation of 10. This will be a measure of the true ability of people on some characteristic. You also created two random error scores having a mean of zero (remember that we always assume that errors have zero mean) and a standard deviation half that of the true score. These errors will be used to construct two separate "tests":

```
MTB> Add C1 C2 C4.
MTB> Add C1 C3 C5.
```

Thus, you have created two tests of some ability. The tests are related or correlated because they share the same true score (i.e., they measure the same true ability) Name the five variables you have created so far:

```
MTB> Name C1= ''true' C2 = 'x error' C3 = 'y error' C4 = 'X' C5 = 'Y'
```

Check on the distributions:

```
MTB> Describe C1-C5
```

You should find that the mean of the TRUE variable is near 50, the means of the two error variables are near 0 and the means of x and y are near 50. Now look at the distributions of the two tests to see if they appear to be normal in shape:

```
MTB> Histogram C4
MTB> Histogram C5
```

The graphs should look like bell-shaped curves. You should also look at the correlation between x and y:

```
MTB> Correlation C4 C5
```

and at the bivariate distribution:

```
MTB> Plot C5 * C4;
SUBC> Symbol 'x'.
```

Up to now, this is pretty much what you did in the first exercise. Now let's look at regression to the mean. Remember that we said that regression will occur whenever we select a group asymmetrically from a distribution. Let's say that the X test is a measure of math ability and that we would like to give a special math program to all those children who score below average on this test. We would then like to select a group of all those scoring below the mean of 50:

```
MTB> Copy C4 C5 C1-C3 C6-C10;
SUBC> use C4 = 0:50.
```

Be sure to type these commands in exactly as written. In words, what you have done is to "COPY all those with scores between 0 and 50 on variable C4 (the X test) and to also COPY the scores in C5 (the Y test) and C1 through C3 for those cases; and put the copied cases in C6 through C10 respectively." Now, name the new columns:

```
MTB> Name C6='New x' C7='New y' C8='New t' C9='New xe' C10='New ye'
```

Notice that the order of the variables has been rearranged. The X test score for the group you chose is in C6 and the Y test score is in C7.

Now, assume that you never gave your math program (your school district lost all funding for special training - sometimes simulations can be realistic!) but that you measured your selected group later on a similar math test, the Y test. Look at the before-and-after means for your selected group:

```
MTB> Describe C6 C7
```

It looks like your group improved slightly between the two tests! Does this mean that your program might not have been necessary? Is this improvement merely normal maturation in math ability? Of course not! All you are witnessing is regression to the mean. You selected a group that consisted of low-scorers on the basis of the X test. On any other measure which is imperfectly related to the X test this group will appear to score better simply because of regression to the mean. Look at the distributions of the selected group:

```
MTB> Histogram C6
MTB> Histogram C7
```

Notice that the distribution of the X test for this group looks like one half of a bell-shaped curve. It should because you selected the persons scoring on the lower half of the X test. But the Y test distribution is not as clearly cut as the X test. Why not? Persons have the same true score on both tests (remember that you added in the same true score to the X and Y test). It must have something to do with the errors then. Obviously, since every person has the same true score on both tests, and since persons appeared on average to score a little higher on the Y test than on the X test, we should expect to see more negative errors on the X test (i.e., x errors) than on the Y test. Let's see if that is true. You can use the SIGN command to see how many negative, zero, and positive values a variable has:

```
MTB> Sign C9
MTB> Sign C10
```

There should be a few more negative errors for the X test and a few more positive errors for the Y test. Are there? At this point you should stop to reflect on what you've done. You created two tests that measure the same ability. The tests are imperfectly measured (i.e., they have error). You then selected an asymmetrical sample on the basis of scores on one of the tests - you selected the "low" scorers on the X test. Even though you didn't do anything to this group, when you measured them again on the Y test you found that they appear to improve slightly. If you worked in a school district, people might begin to question your suggestion that those children needed special training. Let's say that you decided to show them that the apparent gain in this group really isn't accurate. You decide that you will look at the change between the X and Y test for those children who score above the mean:

```
MTB> COPY C4 C5 C1-C3 C6-C10;
SUBC> use C4 = 50:100.
```

Now you look at the means for this above average group on the X and Y tests:

```
MTB> Describe C6 C7
```

What happened? It appears that the above average group lost ground between the X and Y tests. Now your critics are really convinced (or should we say confused?). They argue that the low scorers improved just fine without your special math program but that the high scorers were the ones who lost ground. Maybe they say, you should be giving your program to the high scorers to help prevent further decline.

What is going on here? What you have witnessed is a statistical phenomenon called regression to the mean. It occurs because we have imperfect measurement. Recall that a child's true ability in math is indicated by the true score. The tests partially show us the true score but they also have error in them. For any given child the error could work for or against them - they could have a good day (i.e., a positive error) or a bad one (i.e., negative error). If they have a bad day, their test score will be below their true ability. If they have a good day, the test score will be above their true ability (i.e., they got lucky or guessed well). When you selected a group that was below the overall average for the entire population of 500 children, you chose a lot of children who really did have below average true ability - but you also chose a number of children who scored low because they had a bad day (i.e., had a negative x error). When you tested them again, their true ability hadn't changed, but the odds that they would have as bad a day as the first time were much lower. Thus, it was likely that on the next test (i.e., the Y test) the group would do better on the average.

It is possible that sometimes when you do the above simulation, the results will not turn out as we have said. This is because you have selected a group that is really not too extreme. Let's really stack the deck now and see a clear cut regression artifact. We will choose a really extreme low scoring group:

```
MTB> COPY C4 C5 C1-C3 C6 C10;
SUBC> use C4 = 0:40.
```

We have selected all those who scored lower than 40 on the X test. Look at the distributions:

```
MTB> Histogram C6
MTB> Histogram C7
```

Here, the X test distribution looks like it was sharply cut, the Y test looks much less so. Now, look at the means:

```
MTB> Describe C6 C7
```

Here we have clear regression to the mean. The selected group scored much higher on the Y test than on the X test.

To get some idea of why this occurred look at the positive and negative values of the errors for the two tests:

```
MTB> Sign C9
MTB> Sign C10
```

There should be far more negative values on the x error than on the y error (new x_e and new y_e).

We can predict how much regression to the mean will occur in any case. To do so we need to know the correlation between the two measures for the entire population (the first correlation you calculated above) and the means for each measure for the entire population (in this case, both means will be near 50). The percent of regression to the mean is simply 100 (1-r) where r is the correlation. For example, assume that the correlation between the X and Y tests for all 500 cases is .80 and that the means are both equal to 50. Further, assume you select a low scoring group from the X test and when you look at the mean for this group you find that it equals 30 points. By the formula you would expect this mean to regress 100 (1-.8) or 20% of the way back towards the mean on the other measure. The mean that we would expect on the other measure (the Y test) in the absence of regression is the same as for the X test -- 30 points. But with regression we expect the actual mean to be 20% closer to the overall Y test mean of 50. Twenty percent of the distance between 30 and 50 would put the Y test mean at 34 and it would appear that the group gained four points, but this gain would instead be due to regression. Now, assume the same situation, but this time assume the correlation between the X and Y test is .50. Here we would expect that there would be 100 (1-.5) = 50% regression to the mean and if the group which was selected had a 30 on the X test we would expect them to get a 40 on the Y test just because of the regression phenomenon.

There are several variations on this basic simulation that you might want to try:

- You should vary the strength of the correlation between the X and Y tests. You can do this by varying the standard deviations of the x and y errors (in the original three Random/Normal statements). You might want to try the simulation with the error variances very small (Random 500 C2; Normal 0 5., for example) or you could simply lower the variance of the true score (Random 500 C1; /Normal 50 1). Whichever you do, you should realize that you will be affecting the range of the scores that you obtain on the X and Y tests (the smaller the variances, the smaller the range). Make sure that when you do a copy statement you are actually choosing some cases. You should verify for yourself that the higher the correlation between X and Y the lower the regression, and vice versa. You can even simulate the extreme cases. To simulate a perfect correlation, set both X and Y equal to the true score only (e.g., Let C4=Cl). To simulate a zero correlation, set X and Y equal to their respective error terms (and no true score).
- Regression to the mean occurs in two directions. Try selecting persons based on extremes on the Y test and verify that they regress toward the mean on the X test. While it might seem obvious to you now that regression will occur, it has not always been so obvious in practice. Consider a school district that has conducted a pre-post evaluation of a program and would like to look more closely at the results. They might for example, want to look at how the children who scored low or high on the posttest had done on the pretest. They might look at a group of high posttest scorers and would find that these children did "worse" on the pretest. Similarly, they might look at the low posttest scorers and they would find that these children had done

"betterÓ on the pretest. They might conclude that children who did fairly well on the pretest were hurt by their program while poor pretest scorers were helped by it. But this kind of pattern in the results occurs whether a program is given or not - these evaluators would be basing their conclusions entirely on regression artifacts. Try to look at regression in both directions - from the X to the Y test, and vice versa.

Downloading the MINITAB Commands

If you wish, you can download a text file that has all of the MINITAB commands in this exercise and you can run the exercise simply by executing this macro file. To find out how to call the macro check the MINITAB help system on the machine you're working on. You may want to run the exercise several times -- each time will generate an entirely new set of data and slightly different results. Click here if you would like to download the MINITAB macro for this simulation.

```
GMACRO
REGMEAN
echo
Random 500 C1;
Normal 50 10.
Random 500 C2;
Normal 0 5.
Random 500 C3;
Normal 0 5.
Add C1 C2 C4.
Add C1 C3 C5.
Name C1= 'true'  C2 = 'x error'  C3 = 'y error'  C4 = 'X'  C5 = 'Y'
Describe C1-C5
Histogram C4
Histogram C5
Correlation C4 C5
Plot C5 * C4;
Symbol 'x'.
Copy C4 C5 C1-C3  C6-C10;
use C4 = 0:50.
Name C6='New x'  C7='New y'  C3='New t' C9='New xe'   C10='New ye'
Describe C6 C7
Histogram C6
Histogram C7
Sign C9
Sign C10
COPY C4 C5 C1-C3 C6-C10;
use C4 = 50:100.
Describe C6 C7
COPY C4 C5 C1-C3 C6-C10;
use C4 = 0:40.
Histogram  C6
Histogram C7
```

```
Describe C6 C7
Sign C9
Sign C10
ENDMACRO
```

Applications of Simulations in Social Research

There are a number of ways in which the simulation exercises can be useful in program evaluation contexts. First, they provide a powerful teaching tool (Eamon, 1980; Lehman, 1980). Students of program evaluation can explore the relative advantages of these designs under a wide variety of conditions. In addition, the simulations show the student exactly how an analysis of these designs could be accomplished using real data. Second, the simulations provide a way to examine the possible effects of evaluation implementation problems on estimates of program effect (Mandeville, 1978; Raffeld et al., 1979; Trochim, 1984). Just as NASA explores difficulties in a space shuttle flight using an on-ground simulator, the data analyst can examine the possible effects of attrition rates, floor or ceiling measurement patterns, and other implementation factors. Finally, simulations make it possible to examine the potential of new data analysis techniques. When bias is detected in traditional analysis and analytic solutions are forthcoming, simulations can be a useful adjunct to statistical theory.

Applications For Teaching

Simulations offer several advantages for teaching program evaluation. First, students can construct as well as observe the simulation program in progress and get an idea of how a real data analysis might unfold. In addition, the simulation presents the same information in a number of ways. The student can come to a better understanding of the relationships between within-group pretest and posttest means and standard deviations, bivariate plots of pre-and postmeasures that also depict group membership, and the results of the ANCOVA regression analyses. Second, the simulations illustrate clearly some of the key assumptions that are made in these designs and allow the student to examine what would happen if these assumptions are violated. For instance, the simulations are based on the assumption that within-group- pre-post slopes are linear and that the slopes are equal between groups. The effects of allowing the true models to have treatment interaction terms or nonlinear relationships can be examined directly with small modifications to the simulation program as Trochim (1984) illustrated for the RD design. Third, the simulations demonstrate the importance of reliable measurement. By varying the ratio of true score and error term variances, the student can directly manipulate reliability and show that estimates of effect become less efficient as measures become less reliable. Finally, simulations are an excellent way to illustrate that apparently sensible analytic procedures can yield biased estimates under certain conditions. This is shown most clearly in the simulations on the NEGD. Although the apparent similarity between the design structures of the RE and NEGD might suggest that traditional ANCOVA regression models are appropriate, the simulations clearly show this to be false and thereby confirm the statistical literature in the area (Reichardt, 1979).

Applications for the Study of Design Implementation

The validity of estimates from the simulation exercises contained in this manual depend on how well they are executed or implemented in the field. There are many implementation problems occurring in typical program evaluations--attrition problems, data coding errors, floor and ceiling effects on measures, poor program implementation, and so on--that degrade the theoretical quality of these designs (Trochim, 1984). Clearly, there is a need for improved evaluation quality control (Trochim and Visco, 1985), but when implementation problems cannot be contained, it is important for the analyst to examine the potential effects of such problems on estimates of program gains. This application of simulation is analogous to simulation studies that NASA conducts to try to determine the effects of

problems in the functioning of the space shuttle or a communications satellite. There, an exact duplicate of the shuttle or satellite is used to try to recreate the problem and explore potential solutions. In a similar way, the program evaluator can attempt to recreate attrition patterns or measurement difficulties to examine their effects of the analysis and discover analytic corrections that may be appropriate. The analyst can directly manipulate the models of the problems in order to approximate their reality more accurately and to examine the performance of a design under more varied situations. Such simulations are useful in that they can alert the analyst to potential bias and even indicate the direction of bias under various assumptions.

Applications For the Investigation of New Analyses

One of the most exciting uses of simulation involves the examination of the accuracy and viability of "new" statistical techniques that are designed to address the deficiencies of previous models. There are two reasons why simulations are particularly valuable here. First, the conditions that the analysis will yield unbiased estimates. Second, simulations allow the analyst to examine the performance of the analysis under degraded conditions or conditions that do not perfectly match the mathematical ideal. Thus, simulations can act as a proving ground for new analyses that supplement and extend what is possible through mathematical argument alone.

This application of simulations can be illustrated well by reflecting on the NEGD simulations, where the estimates of program effect were clearly biased. This bias is well know in the methodological literature (Reichardt, 1979) and results from unreliability (measurement error) in the preprogram measure under conditions of nonspecific able group nonequivalence. One suggestion for addressing this problem analytically is to conduct what is usually called a reliability-corrected analysis of Covariance to adjust for pretest unreliability in the NEGD. The analysis involves correcting the pretest scores separately for each group using the following formula:

$$X_{adj} = X + r_{xx}(X_i - X_{mean})$$

where:

X_{adj} = the adjusted or reliability corrected pretest

X_{mean} = the within-group pretest mean

X_i = pretest score for case i

r_{xx} = an estimate of pretest reliability

The analyst must use an estimate of reliability and there is considerable discussion in the literature (Reichardt, 1979; Campbell and Boruch, 1975) about the assumptions underlying various estimates (for example, test-retest or internal consistency). The reader is referred to this literature for more detailed consideration of this issues. The choice of reliability estimate is simplified in simulations because the analyst knows the true reliability (as discussed earlier). This adjusted pretest is then used in place of the unadjusted pretest for the NEGD simulations.

To illustrate the correction, simulations were conducted under the same conditions in the NEGD exercises using the reliability-corrected ANCOVA. It is clear that the reliability corrected NEGD analysis

yields unbiased estimates, thus lending support to the idea that this correction procedure is appropriate, at least for the conditions of these simulations.

Simulations have been used to explore and examine the accuracy of a wide range of statistical analyses for program evaluation including models for adjusting for selection biases in NEGD (Trochim and Spiegelman, 1980; Muthen and Joreskog, 1984); for correcting for misassignment with respect to the cutoff in RD designs (Campbell et al., 1979; Trochim, 1984), and for assessing the effects of attrition in evaluations (Trochim, 1982).

Conclusion

This workbook offers several simulation models that are appropriate for use of three common research designs for evaluating program effects. The logic of these simulations can be easily extended to other relevant research contexts. For instance, many agencies routinely conduct sample surveys to identify needs and target populations, assess services that are provided, and compare agency functioning with the performance of other similar agencies or with some standard. One would construct simulation models for survey instruments for the same reasons that they are constructed for evaluation designs--to improve teaching and general understanding, to explore problems in implementing the survey (such as non response patterns), or to examine the probable effect of various analytic strategies. The key to doing this would again rest on the statistical model used to generate hypothetical survey responses. A "true score" measurement model is useful, at least for simple simulations, but may have to be modified. For instance, assume that one question on a survey deals with client satisfaction with a particular service and that the response is a 7-point Likert-type format where 1=very dissatisfied, 7=very satisfied, and 4=neutral. The analyst could make the assumption that for some sample or subsample the true average response is a scale value equal to 5 points (somewhat satisfied), and that the true distribution of responses is normal around these values, with some standard deviation. At some point, the analyst will have to convert this hypothetical underlying continuous true distribution to the 7-point integer response format either by rounding or by generating normally distributed random integers in the first place. Such a variable could then be correlated or cross-tabulated with other generated responses to explore analytic strategies for that survey. Similar extensions of the models discussed here can be made for simulations of routinely collected management information system (MIS) information, for data for correlational studies, or for time-series situations, among others.

Simulations are assumptive in nature and vary in quality to the degree that the reality is correctly modeled. When constructing a simulation, it is important that the analyst seek out empirical evidence to support the assumptions that are made whenever this is feasible. For instance, it should be clear that the simulations described here could be greatly enhanced if we had more specific data on how much and what type of attrition typically occurs, what type of floor or ceiling effects are common, what patterns of misassignment relative to the cutoff value typically arise for the RD design, what the typical test-retest reliabilities (for use in reliability-corrected ANCOVA) might be, and so on. Although some relevant data will be available in the methodological literature, all of these issues are context specific and demand that the analyst know the setting in some detail if the simulations are to be reasonable.

One way to approach the assumptive nature of the simulation task is to recognize that reality conditions or constraints in the models need to be examined systematically across a range of plausible conditions. This implies that multiple analyses under systematically varied conditions that are based upon principles of parametric experimental design are needed in state-of-the art simulation work. This point is made well by Heiberger et al. (1983:585):

The computer has become a source of experimental data for modern statisticians much as the farm field was to the developers of experimental design. However, many "field" experiments have largely ignored fundamental principles of experimental design by failing to identify factors clearly and to control them independently. When some aspects of test problems were varied, others usually changed as well--often in unpredictable ways. Other computer-based experiments have been ad hoc collection of anecdotal results at sample points selected with little or no design.

Heiberger et al. (1983) go on to describe a general model for simulation design that allows the analyst to control systematically a large number of relevant parameters across some multidimensional reality space, including the sample size, number of endogenous variables, number of "key points" or condition values, matrix eigenvalues and eigenvectors, intercorrelations, least squares regression coefficients, means standard errors, and so on.

Although rigorous, experimentally based simulations are essential for definitive analysis of complex problems, they will not always be feasible or even desirable for many program evaluation contexts. Instead, it is important to recognize that simulations are generally a useful tool that can be used to conduct more definitive statistical studies. However, simulations in program evaluation can provide the analyst with the means to explore and probe simple relevant data structures for the purposes of improving the instruction of research, examining research implementation issues and pilot testing analytic approaches for problematic data.

References

Bradley, D. R. (1977) "Monte Carlo simulations and the chi-square test of independence." Behavior Research methods and Instrumentation 9: 193-201.

Campbell D.T. and R.F. Boruch (1975) "Making the case for randomized assignment of treatments by considering the alternatives: Six ways in which quasi-experimental evaluations tend to underestimate effects," in C.A. Bennet and A.A. Lumsdaine (eds.) Evaluation and Experience: Some Critical Issues in Assessing Social Programs. New York: Academic Press.

Campbell, D.T., C.S. Reichardt, and W. Trochim (1979) "The analysis of fuzzy regression discontinuity design: Pilot Simulations." Unpublished manuscript, Department of Psychology, Northwestern University, Chicago.

Cook, T.D. and D. Campbell (1979) Quasi-experimentation: Design and Analysis Issues for Field Settings. Chicago: Rand-McNally.

Eamon, D.E. (1980) "Labsim: A data driven simulation program for instruction in research and statistics." Behavior Research Methods and Instrumentation 12: 160-164.

Golderger, A.S. (1972) "Selection bias in evaluating treatment effects: Some formal illustrations." Discussion paper, Institute for Research on Poverty, University of Wisconsin, Madison.

Guetzkow, H (1962) Simulation in Social Science. Englewood Cliffs, NJ: Prentice-Hall.

Heckman. J. (1981) "The incidental parameters problem and the initial conditions in estimating a discrete time-discrete data stochastic process." in C.F. Manski and D. McFadden (eds.) Structural Analysis of Discrete Data with Economic Applications. Cambridge, MA: MIT Press.

Heiberger, R.M., P.F. Veleman, and A.M. Ypelaar (1983) "Generating test data with independent controllable features for multivariate general linear forms." Journal of the American Statistical Association 78:585-595.

Lehman, R.S. (1980) "What simulations can do to the statistics and design course." Behavior Research Methods and Instrumentation 12:157-159.

Mandell, L.M. and E.L. Blair (1980) "Forecasting and evaluating human service system performance through computer simulation," pp. 60-67 in American Statistical Association Proceedings, Washington, DC: American Statistical Association.

Mandeville, G.K. (1978) "An evaluation of Title I and model C1: The special regression model." American Education Research Association. Proceedings, Washington, D.C, April.

Muthen, D. and K.G. Joreskog (1984) "Selectivity problems in quasi-experimental studies," in R.F. Conner et al. (eds.) Evaluation Studies Review Annual, Vol. 9 Beverly Hills, CA: Sage.

Raffeld, P., D. Stamman, and G. Powell (1979) "A simulation study of the effectiveness of two estimates of regression in the Title I model A Procedure." American Educational Research Association Proceedings, Washington, D.C. April.

Reichardt, C. (1979) "The design and analysis of the non-equivalent group quasi-experiment." Unpublished doctoral dissertation, Northwestern University, Chicago.

Ryan, T.A., B.L Joiner, and B.F. Ryan (1991) MINITAB Handbook. PSW-Kent Publishing Company Boston.

Trochim, W. (1982) "Methodologically based discrepancies in compensatory education evaluation." Evaluation Review 6, 3: 443-480.

Trochim, W. (1984) Research Design for Program Evaluation: The Regression Discontinuity Approach. Beverly Hills, CA. Sage.

Trochim, W. (1986) Advances in Quasi-Experimental Design and Analysis. San Francisco: Jossey-Bass.

Trochim, W. and J. Davis (1986) "Computer Simulation for Program Evaluation." Evaluation Review Vol. 5. 10 No. 5, October pp. 609-634.

Trochim, W. and C.H. Spiegleman (1980) "The relative assignment variable approach to selection bias in pretest-posttest group designs," p. 102 in Proceedings of the Social Statistics Section, American Sociological Association, Washington, DC, September.

Trochim, W. and R. Visco (1985) "Quality control in evaluation," in DS. Cordray (ed.) Utilizing Prior Research in Evaluation Planning. San Francisco: Jossey-Bass.