DECOMP
A Program for Multiple Standardization and Decomposition

Version 0.51

Professor Steven Ruggles
Social History Research Laboratory
Department of History
267 19th Avenue South
University of Minnesota
Minneapolis, MN 55455

Table of Contents

** These sections are not yet available.

I. Introduction

DECOMP is a general-purpose program for multiple direct standardization and decomposition. The simpler forms of direct standardization and decomposition are frequently used by demographers, but the more sophisticated versions of these methods are rarely employed, chiefly because the necessary computer programming is onerous. This software will make these powerful analytic tools easily accessible to researchers.

I will forgo a detailed explanation of the methods, but in the course of explaining how to use the program, I will make some general comments about how to interpret the results. Readers unfamiliar with the techniques should refer to Prithwis Das Gupta, "A General Method of Decomposing a Difference Between Two Rates into Several Components," Demography 15:1 (1978), 99-111; Evelyn M. Kitagawa, "Components of a Difference Between Two Rates," Journal of the American Statistical Association 50 (1955), 1168-1194; and Edwin D. Goldfield, "Appendix B: Methods of Analyzing Factors of Labor Force Change," pp. 219-236 in John D. Durand, The Labor Force in the United States: 1890-1960 (New York, 1948). DECOMP follows Das Gupta's approach to decomposition. An easily understandable description of the basic methods of standardization can be found in Henry S. Shryock and Jacob Siegel, The Methods and Materials of Demography (Condensed Edition, San Diego, 1976). For an application of multiple standardization, see U.S. Bureau of the Census, Sixteenth Census of the United States: 1940. Differential Fertility, 1910 and 1940. Standardized Fertility Rates and Reproduction Rates (Washington, D.C., 1944). An example of Das Gupta's method of decomposition can be found in Steven Ruggles, "The Demography of the Unrelated Individual, 1900-1950," Demography 25:4 (1988).

Getting Started

DECOMP is designed to run on a PC-compatible microcomputer with at least 512k of memory. A hard disk is recommended for all but the simplest of problems. You must also have some free disk space for a temporary workfile; theoretically, the program can use as much as 224k of work space, but for most problems about 50k should be sufficient.

To run the program, you must first set up a command file using an ASCII editor or the non-document mode of a word processor. The command file will contain instructions to define the data set, carry out any needed data transformations, and specify the particular standardizations and decompositions.

Installing the program is easy: just copy the files on the DECOMP diskette to your hard disk or a to a backup floppy disk. If you have a hard disk, you may want to create a decomp subdirectory and alter the PATH command in your autoexec.bat file, so you can run the program from any drive and directory.

Start the program by typing the command DC at the system prompt. The program will then ask you for the name of your command file. If you are running the program from a floppy drive system, you may remove the program diskette at this time and replace it with a a disk containing data or your command file. By default, the results will appear in a file called 'decomp.lis'.

Data Requirements

The input data for DECOMP must be contained in an ASCII file consisting of non-negative numbers in column format, with one record per case. Although most social science data sets are organized this way, some are not. If the data set includes negative numbers, alphabetic characters, or is free-format or has multiple records per case, you will have to convert it using another program before it can be read into DECOMP. In addition, DECOMP will not read data beyond 200 columns, so data sets with unusually long records will also have to be converted. General purpose statistical packages such as SPSS/PC+ or SAS-PC can perform all these conversions easily. If your data are in column format but contain alphabetic characters or negative numbers that you do not intend to use, DECOMP will skip over the offending columns, so conversion is not necessary. DECOMP is primarily oriented to analysis of individual-level or household-level data files. Few aggregate data files are appropriate for multiple standardization or decomposition analysis, because they are rarely broken down by enough variables to make it worthwhile. However, DECOMP can handle aggregate data through use of its WEIGHT command, described below. The maximum number of cases is five million.

Command Structure

In general, DECOMP commands are very similar to those used in the statistical analysis program SPSS/PC+. As in SPSS/PC+, all commands must be terminated by a period. If you leave the period off the end of a command, the subsequent command will be ignored or misinterpreted. In addition, the program will not read commands that extend beyond 80 columns; if you need more than 80 columns, continue the command on the next line. You may use as many lines as you wish, as long as each command uses no more than 500 meaningful characters. DECOMP ignores extra spaces, except that commands should begin in the first column, and it is not sensitive to case.

II. Basic DECOMP Commands

To run a decomposition or standardization, you need at least three basic commands: (1) a DATA LIST command that identifies the data file, variable names, and location of the variables; (2) a MAKETAB command that constructs a multi-dimensional crosstabulation needed for both standardization and decomposition; and (3) either a STANDARDIZE or a DECOMPOSE command that defines your particular analysis. Most of the time, you will probably use some of the additional DECOMP commands: SELECT IF, RECODE, COMBINE, WEIGHT, or SET. Since these are not essential, however, I will defer discussion of them until later sections.

For each command, the syntax is given in the following form:

- -- Keywords are shown in capitals
- -- Specifications supplied by the user are given in lower case
- -- options are shown in square brackets []

The DATA LIST command

Overview: Defines the characteristics of your data file. At least one DATA LIST command is required for every run. Ordinarily, the DATA LIST command should appear first in your command file (although you may put a SET command first). The DECOMP version of this command is a subset of that used in SPSS/PC+.

Syntax:
```
DATA LIST FILE='filename'
/varname columns varname columns varname columns.
```

where:

filename is the DOS filename of your data file, including the drive and path if the data are not located in the current DOS directory;
varname is the name of each variable to be used by DECOMP;
columns is the range of columns for each variable.

The filename must appear within single quotes. It may include specifications for disk drive and subdirectory, as long as the total length does not exceed 35 characters. Variable names may be up to 10 characters long. The columns should either consist of a single integer between 1 and 200, or a range separated by a dash. Column ranges may not exceed 8 columns. Up to 30 variables may be specified. If your data includes real numbers (numbers with decimal points), don't worry about it here; just give the total range of columns.

Example:
```
DATA LIST
FILE='c:\census\pu1900.dat'
/age 19-21 sex 13 mstat 22 chborn 25-26 race 12
rectype 70.
```

The MAKETAB command

Overview: The MAKETAB must appear after the DATA LIST command and before the STANDARDIZE or DECOMPOSE commands. MAKETAB specifies the dependent variable and other variables available for analysis, and creates a table with up to five dimensions containing the number of cases and the value of a dependent variable for each combination of characteristics in the population. These tables are generally too complex for humans to read (they can contain up to 56,000 cells), but they are necessary for the analysis. Therefore, the results of the MAKETAB command are stored in a temporary binary file on disk until they are called up by a STANDARDIZE or a DECOMPOSE command. As an option, you may write the table to an ASCII disk file for later analysis with another program.

The dependent variable must either be dichtomous or interval scale. All the other variables specified in the MAKETAB command must be categorical. In general, you should keep the number of categories of these variables as small as feasible without losing important detail. The product of the number of categories for the other variables cannot exceed 28,000. In most cases, you should keep the analyses much smaller than that, since few data sets are large enough to support such detail. The dependent variable may be dichotomous if you are analyzing a rate or percentage, or it may be an integer or a real number if you are analyzing means.

Syntax:
```
MAKETAB DEPENDENT=varname[(n)]
/VARIABLES=varname(min,max) varname(min,max)
varname(min,max) varname(min,max) varname(min,max)
[/WRITE TABLE].
```

where:

n is the number of decimal places to the right of the decimal point for the dependent variable. This need only be specified when the dependent variable is a real number.

min,max are the minimum and maximum values for each variable, separated by a comma

All variable names must appear exactly as they were defined in the DATA LIST command. Except for the dependent variable, the minimum and maximum values of each variable must be specified. The minimum allowed value is zero; there is no maximum, but values greater than 999 may not be displayed properly on the output tables. No more than five variables in addition to the dependent variable may be specified (if your analysis requires more than five variables, see the COMBINE command).

Examples:
```
MAKETAB DEPENDENT=chborn /VARIABLES=age(15,44) mstat(1,3)
race(1,2).

MAKETAB
    DEPENDENT=wagerate(2)
              /VARIABLES=educ(5,14) occ(1,11) agegrp(1,15)
                      sex(1,2) race(1,2).
```

In the second of these examples, the variable wagerate is expressed in dollars and cents, and therefore there are two digits to the right of the decimal point, identified by the (2) following the variable name. It does not matter whether or not a decimal point actually appears in the data; the program will interpret the two right columns of wagerate as cents in any case. If the (2) were left out, the decimal point would be ignored, and wagerate would be expressed in cents.

WRITE TABLE subcommand. As an option, you may write the working table to an ASCII disk file for later analysis by another program. In fact, DECOMP can serve as a general-purpose pretabulation program to speed up other software. For a discussion of this, see the section entitled "Using DECOMP to Pretabulate Data Sets."

```
Example:     MAKETAB DEPENDENT=foreign
             /VARIABLES=region(0,9) age(0-99) sex (0,1) marstat (1,4)
             metro(1,2)
             /WRITE TABLE.
```

When the /WRITE TABLE subcommand is issued, the program will automatically generate a codebook to read the table. By default, the codebook will appear in the 'decomp.lis' file, and the table will appear in the 'decomp.tab' file. (You can override these defaults by using a SET command.) The following codebook was created with the MAKETAB command shown above.

The table is written to file DECOMP.TAB
using the following format:

Variable Name	Columns
REGION	1- 1
AGE	3- 4
SEX	6- 6
MARSTAT	8- 8
METRO	10-10
Mean of dependent	12-19
Number of cases	21-23

The mean of the dependent variable is written with four columns to the right of the decimal point; the other variables are written as integers, except that the number of cases will be written as a real number when necessary because of a weighted data set.

The STANDARDIZE command

Overview: The STANDARDIZE command must appear after a MAKETAB command. It specifies what groups are to be compared and what variables should be controlled. Options also allow you to specify what standard population should be employed, in what format the results are to be presented, and whether excluded cases should be written to a file for later analysis.

Syntax:
```
STANDARDIZE
/BREAKDOWN=varname, varname, varname, varname
/CONTROL=varname, varname, varname, varname
[/STANDARD=TOTAL]
[/STANDARD=AVERAGE]
[/STANDARD=CATEGORY(n)]
[/FORMAT=PERCENTS]
[/FORMAT=DEVIATIONS]
[/WRITE EXCLUDED CASES].
```

All variables mentioned in the STANDARDIZE command must be specified in the preceding MAKETAB command. The BREAKDOWN and CONTROL subcommands are required; all the others are optional. The BREAKDOWN subcommand specifies the variable(s) that define the groups to be compared, and the CONTROL subcommand specifies the variable(s) representing characteristics to be standardized by. STANDARDIZE allows a maximum of five BREAKDOWN variables and four CONTROL variables, except that five CONTROL variables may be specified when there are five identical BREAKDOWN variables. You must specify the BREAKDOWN variable(s) before the CONTROL variable(s).

Example: The following command could be used to compare the fertility of blacks and whites, controlling for their age structure.

```
STANDARDIZE
/BREAKDOWN=race
/CONTROL=age.
```

BREAKDOWN subcommand: STANDARDIZE allows you to do up to five standardizations with a single command. The following command would successively compare whites and blacks, income groups, educational groups, and regions:

Example:
```
STANDARDIZE
/BREAKDOWN=race, income, educ, region
/CONTROL=age.
```

CONTROL subcommand: You can also standardize up to four characteristics simultaneously, as in the following example.

Example: `STANDARDIZE /BREAKDOWN=race /CONTROL=age, income, educ, region.`

Sample run #1: Before describing the various options of the STANDARDIZE command, let me give a example of a complete DECOMP run with real results. Figure 1 shows a job to read several variables from an extract of the women of childbearing age in the 1900 Public Use Sample of the U.S. census and standardize children-ever-born to native and foreign-born women, controlling for age and marital status.

The three necessary commands are echoed to the output file automatically. The DATA LIST command instructs the program to read four variables from the file FEM00.DAT on the E: drive. MAKETAB creates a working table with CHBORN (children-ever-born) as the dependent variable, broken down by NATIVE (native vs. foreign born), AGE (by single years), and MARSTA (marital status). Finally, the STANDARDIZE command directs the program to compare the CHBORN of native- and forign-born women, controlling for age and marital status.

Before displaying the results, DECOMP provides some information about the run. First, it identifies the dependent variable, CHBORN. Second, it tells what standard population was used for the analysis, and third, what format the results are expressed in. The standard population and output format are controlled by the STANDARD and FORMAT subcommands, described below; for this run, the defaults were used. Next, the listing identifies the BREAKDOWN and CONTROL variables.

The presentation of results begins by displaying the overall mean of the dependent variable for all cases, and the number of cases used in the analysis. This run used some 23,000 cases. This may seem a high number for a microcomputer, but DECOMP is pretty fast; this job took 29 seconds on a IBM Model 80-111.

The results are expressed in tabular form. The categories of NATIVE are given on the left of the table. DECOMP does not support labels for the breakdown categories, so you just have to remember what they mean. In this case, NATIVE category 1 refers to native-born women, and category 2 identifies foreign-born women. The next column displays the unstandardized means for each category. In this case, you can see that foreign-born women had on average about one more child that native-born women. The third column shows the standardized means, which indicate what the mean number of children-ever-born in each group would be if each group had the same distribution of marital status and age as the population as a whole. The result shows that if native- and foreign-born women were identical in age structure and marital status, there would have been a relatively small difference in children born. Finally, the right-hand side of the table shows the number of cases in each breakdown category.

Figure 1

```
DATA LIST FILE='E:FEM00.DAT'
/AGE 1-2 MARSTA 3 CHBORN 4-5 NATIVE 6.
MAKETAB
/DEPENDENT=CHBORN
/VARIABLES=NATIVE(1,2) AGE (15,44) MARSTA (1,3).

STANDARDIZE
/BREAKDOWN=NATIVE
/CONTROL=AGE,MARSTA.
```

M U L T I P L E S T A N D A R D I Z A T I O N A N A L Y S I S

Standardization of CHBORN

Standard is the total combined population
Results expressed as means of dependent variable 3+ 3

Breakdown by NATIVE

Controlling for:
AGE
MARSTA

Grand Mean: 2.0249 Total cases included: 23109

Category of NATIVE	Unstandardized Mean	Standardized Mean	N
1	1.8799	1.9758	19467
2	2.7998	2.1915	3642

WARNING: 45 Cases excluded because of empty Cells
(.2 percent of total)

At the bottom of the table there is a WARNING that 45 cases were excluded because of empty cells. The standardization will not work if the populations being compared do not have comparable control characteristics. That is, in this instance, that for every combination of age and marital status in one population, there must be someone with the same characteristics in the other population. If the breakdown groups are not strictly comparable, then everyone with the offending set of characteristics is removed from the analysis, and a warning is generated to inform the analyst. As it happens, there are 10 married 15-year native-born women in the 1900 Public Use Sample, and no married 15- year old women among the foreign-born. Thus, the program threw away those 10 native- born women to make the two populations exactly comparable. The other 35 cases excluded were widowed native-born women between 16 and 22, excluded because there were no foreign-born women with those characteristics.

In this instance, the excluded cases present no great problem, because they represent a very small proportion of the population. However, it is easy to come up with designs that exclude a large proportion of cases. For further discussion of how to deal with excluded cases, see the WRITE EXCLUDED subcommand and the section entitled "Dealing With Excluded Cases."

STANDARD subcommand. If you do not specify otherwise STANDARDIZE will assume that the standard population should be the combined population of all categories of the breakdown variable. Thus, in the previous example, the results would show what the fertility of whites and blacks would be if each group shared the characteristics of the population as a whole. If you want to use an alternate standard, you will need to specify it with the STANDARD subcommand. In the next example, I have specified that the standard should be the average of the proportions of whites and blacks in each age group:

```
Example:          STANDARDIZE
                  /BREAKDOWN=race
                  /CONTROL=age
                  /STANDARD=AVERAGE.
```

Selection of an inappropriate standard population can distort the results, and the average of the breakdown categories is the standard least likely to cause problems. The problem with this approach is that the standard population will be different every time you modify the breakdown categories. Suppose you want to make two standardized tables, the first broken down by race and the second broken down by occupational group. If you use the STANDARD=AVERAGE subcommand, the two tables will be based on different standards, and so will not be strictly comparable. The default standard, STANDARD=TOTAL, will avoid this problem, since both tables will use the entire population as the standard.

You can also use one of the breakdown categories as the standard. In the previous example, if you wanted the standard population to be whites, you would specify that as follows:

```
Example:   STANDARDIZE
           /BREAKDOWN=race
           /CONTROL=age
           /STANDARD=CATEGORY(1).
```

FORMAT subcommand. The Format subcommand determines how the results of the STANDARDIZE command are presented. If no FORMAT subcommand is given, the results will be presented as means of the dependent variable, as shown in Figure 1. If the dependent variable is a dichotomous variable that takes the values 0 and 1, you may specify the FORMAT=PERCENTS subcommand, and the table will be presented in percentage terms. You may also express the results as deviations from the grand mean, by using the FORMAT=DEVIATIONS subcommand. This will produce a table similar to a Multiple Classification Analysis (see the section entitled "Standardization and MCA Compared").

Examples:
```
STANDARDIZE
/BREAKDOWN=region
/CONTROL=income,agegrp
/FORMAT=PERCENTS.

STANDARDIZE
/BREAKDOWN=region,race,sex,age
/CONTROL=region,race,sex,age
/FORMAT=DEVIATIONS
/STANDARD=AVERAGE.
```

WRITE EXCLUDED subcommand. As noted in my discussion of sample run #1, STANDARDIZE will automatically discard cases when the combination of control characteristics does not appear in one or more of the breakdown categories. Thus, for example, if AGE is a control variable and one of the breakdown categories has no 57 year- olds, all 57 year-olds will be excluded from the analysis. There is a variety of strategies to avoid excluding a substantial percentage of cases; these are discussed in the section entitled "Dealing with Excluded Cases." Frequently, it helps to analyze the excluded cases to determine which combinations of characteristics are causing the problem. The WRITE EXCLUDED subcommand will create a file called 'decomp.tab' containing a summary of the excluded cases. The breakdown variable is written first, then each of the control variables, then the mean of the dependent variable for that combination of characteristics, and finally the number of cases involved. Like the WRITE TABLE subcommand of the MAKETAB command, the WRITE EXCLUDED subcommand will write a codebook to your output file.

Example:
```
STANDARDIZE
/BREAKDOWN=region
/CONTROL=income
/WRITE EXCLUDED CASES.
```

Sample Run #2: An example of a DECOMP run using some of the STANDARDIZE subcommands is shown in Figure 2, which standardizes the percentage of unrelated individuals (persons residing without family) across three different regions of the United States in 1940, controlling for age group, sex, marital status, and residence in a metropolitan area. The analysis is based on an extract of the 1940 Public Use Sample, and the categories of the REGION variable are Northeast (1), Midwest (2), and South (3). The standard population is specified as the Northeast, the results are expressed as percentages, and the excluded cells are written to disk. Note that the figure for the Northeast is unaffected by the standardization, because the standard population is the Northeast. Overall, the unstandardized numbers show the highest frequency of unrelated individuals in the Northeast, while the standardized percentage is highest in the South.

Figure 2

```
DATA LIST FILE='D:\CENS40\CEN41.DAT'
/REGION 14 UNREL 24 METRO 85 AGEGRP 28 SEX 30 MARST 32.
MAKETAB
/DEPENDENT=UNREL
/VARIABLES=REGION(1,3) AGEGRP (0,9) SEX(0,1) MARST(1,3) METRO(1,2).
STANDARDIZE
/BREAKDOWN=REGION
/CONTROL=AGEGRP,SEX,MARST,METRO
/STANDARD=CATEGORY(1)
/FORMAT=PERCENTS
/WRITE EXCLUDED CASES.
```

M U L T I P L E S T A N D A R D I Z A T I O N A N A L Y S I S

Standardization of UNREL

Standard is breakdown category 1
Results expressed as means of dependent variable

Breakdown by REGION

Controlling for:
AGEGRP
SEX
MARST
METRO

Grand Mean: 9.2701 Total cases included: 5987

Category of REGION	Unstandardized Mean	Standardized Mean	N
1	10.8787	10.8787	1912
2	8.5698	10.5098	1797
3	8.4723	14.0294	2278

WARNING: 77 Cases excluded because of empty Cells
(1.3 percent of total)

The DECOMPOSE command

Overview: The DECOMPOSE command must appear after a MAKETAB Command. It specifies two groups to be compared, which factors to analyze, and whether excluded cases should be written to a file for later analysis. The decomposition then shows how much of the difference between the two groups can be attributed to each factor.

Syntax:
```
DECOMPOSE
/COMPARE=varname(a:b)
/FACTORS=varname varname varname varname
[/WRITE EXCLUDED].
```

where:
a and b represent two categories of a variable to be compared.

All the variables used in a DECOMPOSE command have to be specified in the preceding MAKETAB command. You may have only one comparison variable and up to four factors.

Example:
```
DECOMPOSE
/COMPARE=year(4:8)
/FACTORS=age,sex,income,education.
```

WRITE EXCLUDED subcommand. This works the same as the WRITE EXCLUDED subcommand of the STANDARDIZE command. Also, see the section on "Dealing With Excluded cases."

Sample Run #3: The easiest way to explain how to use the DECOMPOSE command is by illustration. Figure 3 shows a decomposition of the difference in yearly earnings between whites and non-whites in 1940, based on an extract of the 1940 Public Use Sample. The analysis is restricted to wage and salary workers between the ages of 15 and 65. The factors included in the analysis are age, sex, education, and region.

The output begins with a descriptive table. I have found through experience that such a table is often critical for understanding decomposition results. The left columns of the descriptive table show the mean earnings of whites (labeled 0) and non-whites (labeled 1), broken down by each category of each factor. The right two columns of the descriptive table simply give the frequency distribution of each factor for the two populations. The number of cases in each group, and the number of excluded cases, appears on the bottom of the table.

The categories of the factors are as follows:

Age: 10-year groups, 15-24 through 55-64
Sex: 0=male, 1=female
School: 1=less than fifth grade, 2=less than eighth grade, 3=eighth grade; 4=some high school, 5=completed high school, 6=beyond high school
Region: 1=Northeast, 2=Midwest, 3=South, 4=West

Figure 3

```
DATA LIST
FILE='D:\CENS40\CENWB.DAT'
/EARN 96-99 SCHOOL 136-137 REGION 13-14 RACE 31 SEX 30 AGEGRP 28.

MAKETAB
/DEPENDENT=EARN
/VARIABLES=AGE(1,5) SEX(0,1) RACE(0,1) REGION(1,4) SCHOOL(1,6).

DECOMPOSE
/COMPARE=RACE(0:1)
/FACTORS=AGE,SEX,SCHOOL,REGION.
```

Figure 3 (Continued)

D E C O M P O S I T I O N A N A L Y S I S

Decomposition of EARN
comparing RACE categories 0 and 1

A. Descriptive Table

Factor/category	Mean of EARN 0	Mean of EARN 1	Population distribution 0	Population distribution 1
AGEGRP				
1	530.9373	287.5460	23.5	23.5
2	1028.6370	456.5038	29.6	32.4
3	1344.5250	472.5162	22.0	24.3
4	1287.4860	527.7690	17.0	13.6
5	1154.5790	617.6260	8.0	6.1
SEX				
0	1153.8170	511.9719	72.5	63.1
1	722.2720	317.1135	27.5	36.9
SCHOOL				
1	776.1417	349.5070	7.1	31.5
2	836.1341	410.2891	13.4	30.8
3	956.9988	516.2941	25.3	13.4
4	971.1540	494.3721	19.3	12.7
5	1034.2590	513.0548	22.2	7.2
6	1638.1420	786.7111	12.8	4.4
REGION				
1	1181.3280	619.0219	37.5	13.5
2	945.9686	631.4254	26.3	11.2
3	918.2570	357.2372	26.1	70.4
4	1027.0230	697.9700	10.1	4.9
Total	1035.2690	440.1114	100.0	100.0
N	3400	2031	3400	2031
Excluded cases	169	43	169	43

Figure 3 (Continued)

B. Decomposition Table

	Components of Difference	Index of Difference
Total population difference	595.1577	100.0
Effect of factor AGEGRP	10.3728	1.7
Effect of factor SEX	36.6327	6.2
Effect of factor SCHOOL	119.5348	20.1
Effect of factor REGION	87.5355	14.7
Combined effect of factors	254.0757	42.7
Rate effect	341.0822	57.3

Overall, whites earned an average of $1035 in 1940, compared with only $440 for non- whites. As one would expect, among both whites and non-whites earnings increased with schooling and age, females earned dramatically less than males, and people in the South and Midwest earned less than those in the Northeast. As the frequency distributions show, there were significant differences in the characteristics of white and nonwhite earners. Most notably, a higher percentage of nonwhites were female, nonwhites had less education than whites, and they were overwhelmingly concentrated in the South. One would expect, therefore, that if there had been no differences between whites and non-whites in terms of age, sex, education, and region, the differences in earnings would have been a good deal smaller.

This conclusion is supported by part B of the output, the decomposition table. Here, we see the total difference in earnings between whites and non-whites broken down into the effects of each factor, the combined effect of factors, and the rate effect. The two columns shown show essentially the same thing, except that the numbers on the right are expressed as percentages of the total difference between the two populations.

The rate effect shown at the bottom represents that portion of the difference in earnings that would remain even if the two populations were identically distributed with respect to their characteristics. It is calculated as the sum over all possible combinations of characteristics of the difference in earnings times the average of the proportion of each population with that set of characteristics. The rate effect is equivalent to the difference between whites and non-whites that would remain after simultaneously standardizing by the factors, using the average of the two populations as the standard.

The combined effect of factors represents the amount of difference between the populations that would exist if there were no differences in income, and only the population distribution varied. It is the sum over all combinations of characteristics of the difference in the proportion of the two groups with those characteristics times the average of their income. Note that the sum of the rate effect and the combined effect of factors is the total population difference.

Calculation of the effects of each factor is more complicated. Those who want the nuts and bolts of it must turn to Das Gupta's article mentioned above. Essentially, it works by simultaneously standardizing by every possible combination of factors, so that the effects of each individual factor can be isolated.

In the present example, differences in the composition of the white and non-white populations account for a difference of $242 in earnings, or about 43% of the total difference between the two populations. The most important factors are education and region, and the effects of sex composition are also substantial. Differences in the age distribution of whites and non-whites prove to have little impact on earnings.

Note that the effects of factors need not be positive. A negative effect simply means that if the factor were held constant, the difference between the two populations would grow.

III. Data Transformation and miscellaneous commands

DECOMP includes a variety of additional commands to simplify analysis and add flexibility. Two of these commands -- RECODE and SELECT IF -- allow the user to carry out simple data transformations. The COMBINE command is used to control more than four variables, or when you need a BREAKDOWN by more then one variable simultaneously. The WEIGHT command is necessary to work with weighted or aggregate-level data. Finally, two SET commands allow the user to redirect output.

The RECODE command

Overview: The RECODE command is used to change the coding scheme of a variable on a value-by value basis or for a range of values. It is a simplified version of the RECODE command used in SPSS. RECODES must appear after a DATA LIST and before a MAKETAB command. They will remain in effect until a new DATA LIST command is encountered, even if you have multiple MAKETAB commands. A maximum of approximately 200 input values or THRU operators may appear in a given data pass.

Syntax: RECODE varname (valuelist=newvalue)(valuelist=newvalue)
(valuelist=newvalue) . . . (valuelist=newvalue).

where:
valuelist is a value, series of values, range of values, or combination of these to be recoded;
newvalue is the code to be assigned.

Each set of values must be enclosed in parentheses. Input values are specified first (to the left of the equals sign) followed by a single output value (to the right of the equals sign). Multiple input values can be assigned to a single output value; multiple values in a valuelist must be separated by commas. To indicate a range of input values, use the keyword THRU or a dash. Any input values not specified will remain unmodified.

Examples: RECODE age (0 THRU 9=1)(10 THRU 19=2)(20 THRU 29=3)
(30 THRU 39=4)(40 THRU 49=5)(50 thru 59=6)(60 thru 99=7).

RECODE occup (1,57,62 THRU 155=1)(2 THRU 9,35 THRU 55=2)
(10 THRU 34,56,58 THRU 63=3)(156 thru 999=4).

RECODE race(3 THRU 9=2).

RECODE ethnic (0=1)(1=0)(2,3=9).

RECODE in DECOMP is more limited than it is in SPSS. Only one variable may be specified per RECODE statement; the keywords ELSE, LOWEST, HIGHEST, and MISSING are not allowed in DECOMP.

The SELECT IF command

Overview: The SELECT IF command is used to select cases for analysis. Since there are no special provisions for handling missing data in DECOMP, SELECT IF statements can be used for that purpose. SELECT IF is a much simplified version of the same command in SPSS. The command should appear after a DATA LIST command and before a MAKETAB command, and will remain in effect until a new DATA LIST command is issued. A maximum of 50 SELECT IF commands may appear in a given data pass.

Syntax: SELECT IF (varname operator value).

where:
operator is one of the following relational operators:

Operator	Meaning
EQ	Equal to
LT	Less than
GT	Greater than
LE	Less than or equal to
GE	Greater than or equal to
NE	Not equal to

Examples: SELECT IF (occup NE 999).

SELECT IF (sex EQ 2).

A variable name must always come first, followed by an operator and then a constant. Note that there are no logical operators (such as AND or OR), and therefore only one relational expression can be included in each SELECT IF statement. If you need an AND operator, just issue another select if command.

Example: SELECT IF (age GE 15).
SELECT IF (age LT 45).

These two statements are equivalent to the SPSS command

SELECT IF (age GE 15 AND age LT 45).

There is no easy way to duplicate the effect of an OR operator, although you can trick DECOMP into it by using a COMBINE statement followed by a RECODE and then a SELECT IF; see "Some Tricks" section below.

The COMBINE command

Overview: The COMBINE command creates a new variable by combining two to four existing variables. It serves two important purposes. First, it can be used when you want to have more than four CONTROL variables or FACTORS. Second, a combined variable can be used as a BREAKDOWN variable in STANDARDIZE to create a two-way, three-way or four-way crosstabulation in which every cell is standardized according to a common standard population. In addition, COMBINE can be used to overcome some of the limitations of DECOMP's data transformation capabilities, most notably the lack of IF statements, assignment statements, and logical operators. COMBINE statements must appear between a DATA LIST command and a MAKETAB command, and up to ten COMBINE statements may appear in a given data pass.

Syntax: COMBINE newvar=varname(min,max) varname(min,max)
varname(min,max) varname (min,max).

where:
newvar is the name of the new variable to be created
min,max are the same as in the MAKETAB command, described above

Examples: COMBINE marsexurb=mstat(1,5) sex(0,1) urban(1,3).

COMBINE sexrace=sex(0,1) race(0,1).

The codes for the new variable are described in a conversion table automatically written to the output file. For example, the second of the two examples given above generated the following table:

COMBINE has constructed the new variable SEXRACE
The following codes were assigned:

SEXRACE	SEX	RACE
1	0	0
2	0	1
3	1	0
4	1	1

When a variable created by COMBINE is used in a subsequent MAKETAB command, the minimum and maximum values of the new variable need not be specified in the MAKETAB command, since DECOMP already knows what they will be. Instead of explicitly declaring the minimum and maximum, you can substitute an asterisk enclosed in parentheses. For example, if the variable sexrace was created with COMBINE, the following syntax would be valid for a MAKETAB command:

```
MAKETAB DEPENDENT=income
/VARIABLES=sexrace(*) age(1,99) educ(0,15).
```

That MAKETAB command might be used to compare the incomes of white men, black men, white women, and black women while controlling for age and education. Such an analysis could be carried out with the following STANDARDIZE command:

```
STANDARDIZE
/BREAKDOWN=sexrace
/CONTROL=age,educ.
```

Alternatively, one might use a combined variable like sexrace as a CONTROL variable or a FACTOR when you want to account for more than four variables at once. The following statements show a decomposition of literacy across time periods, with factors for age, sex, race, education, and region.

```
MAKETAB DEPENDENT=lit
/VARIABLES=sexrace(*) agegrp(1,15) educ(0,15)
region(1,4) year(1,2).

DECOMPOSE
/COMPARE=year(1:4)
/FACTORS=sexrace,agegrp,educ,region.
```

The WEIGHT command

Overview: The WEIGHT command is needed if you have weighted data or if you want to use aggregate data. It is essentially the same as the WEIGHT command in SPSS/PC+. WEIGHT should appear after a DATA LIST command and before a MAKETAB command.

Syntax: WEIGHT BY varname[(n)].

where
n indicates the number of places to the right of the decimal point. If no n is indicated, DECOMP assumes you are weighting by whole numbers.

Examples: WEIGHT BY num.

WEIGHT BY wt(4).

In the above example, each record will be weighted (multiplied) by the value of the variable wt, which has four decimal places. It doesn't matter whether or not a decimal point appears in the original data.

The SET LISTING command

Overview: The SET LISTING command is used to change the output (listing) file. (By default, output is routed to the file DECOMP.LIS). A SET LISTING command may appear anywhere in a command file, and you may have as many SET LISTING commands as you wish.

Syntax: SET LISTING='filename'.

where:
filename is the DOS filename your output (listing) file, including the drive and path if you want it to go someplace other than the current DOS directory.

Example: SET LISTING='d:\output\run15.lis'.

The SET RESULTS command

Overview: The SET RESULTS command is used to change the output file used by the /WRITE TABLE and the /WRITE EXCLUDED subcommands of the MAKETAB, STANDARDIZE, and DECOMPOSE commands. (By default, output table is routed to the file DECOMP.TAB). A SET RESULTS command may appear anywhere in a command file, and you may have as many SET RESULTS commands as you wish. If you use than one /WRITE option in a given run, it is recommended that you issue a SET RESULTS command in between; otherwise, you will get more than one table written to the same file.

Syntax: SET RESULTS='filename'.

Example: SET RESULTS='d:\output\exclude.tab'.

IV. The SETUP.CMD file

The SETUP.CMD file is used to customize the DECOMP environment. Whenever the DECOMP program is started, it searches for a file named SETUP.CMD, and executes any commands in that file before doing anything else. DECOMP looks for a SETUP.CMD in the local directory first, and if it is not there the program will search for C:\DECOMP\SETUP.CMD. If the file is still not found, the program looks in any directories named DECOMP on the D: and E: drives.

Although you can put any command in a SETUP.CMD file, there are three special commands that may only appear in a SETUP.CMD, and they are the ones most likely to be used. These commands, described below, allow you to control the prompt for a command file and to suppress page numbering and the screen display. You may also want to put SET LISTING and SET RESULTS in a SETUP.CMD, and if you are using the same data set repeatedly you can also put a DATA LIST and data transformation commands there. However, only the first ten lines of a SETUP.CMD file are read, so additional commands must be placed in an ordinary command file.

The SET PROMPT command

Overview: The SET PROMPT command is used to suppress the opening screen and prompt for a command file when the the DECOMP program is started. When the prompt is set OFF, the user supplies the name of the command file on the command line instead of when the program asks for it. The SET PROMPT command allows you to start the program more quickly, and it is especially valuable if you want to set up DOS batch files to carry out unattended DECOMP runs. The SET PROMPT command may not appear in a regular command file; it will only be recognized if it is part of a SETUP.CMD file.

Syntax: SET PROMPT OFF.

Example: If you have issued the SET PROMPT OFF command in your SETUP.CMD and you wish to run the command file named SAMPLE.CMD, then you should start DECOMP by typing

```
DECOMP SAMPLE.CMD
```

The SET PAGENUMS command

Overview: The SET PAGENUMS command is used to suppress page numbering page breaks, and page headers throughout the output (listing) file. Very often, users will want to edit their output in a word processor before printing, especially because DECOMP has no provision for value labels or variable labels. In such circumstances, the headers and page breaks that are ordinarily inserted by the program can get in the

way. Like the SET PROMPT command, the SET PAGENUMS command can only be issued from within a SETUP.CMD file.

Syntax: SET PAGENUMS OFF.

The SET SCREEN command

Overview: The SET SCREEN command allows you to suppress output to the screen. This results in a slight improvement in speed, and it is especially useful if you are running large DECOMP jobs in the background on a multiprocessing computer. If you SET SCREEN OFF, the SET PROMPT OFF command is issued automatically, so you should specify the command file on the command line. SET SCREEN must be issued from the SETUP.CMD file.

Syntax: SET SCREEN OFF.

V. Error Messages

Error messages appear on the screen and are inserted into the output (listing) file at the point they occurred. With a couple of exceptions, errors halt execution.

Error messages in DECOMP are usually quite specific, but the downside of this is that they are sometimes misleading. The most common source of a misleading message is failure to put a period at the end of a command. When you leave out the period, DECOMP will concatenate the command with its successor. Also, you can get weird messages if you use the wrong kind of slash (\ instead of /) or the wrong kind of single quote mark (` instead of ').

The following pages list all warnings and error messages generated by DECOMP. The messages are left justified, and explanations or other comments are indented.

WARNING: Variable VARNAME outside specified range at case number XXXXXX
Case ignored

Where VARNAME is the first 10 characters of a variable name specified in the preceding MAKETAB command and XXXXXX is a record number in your data set.

This message indicates that a value falles outside the mimimum or maximum range specified in a MAKETAB or COMBINE command. Check your RECODES and SELECT IF statements; if everything seems OK, you may have a data problem.

WARNING: Value over 8 characters truncated

If DECOMP encounters a data value of over 8 digits, excluding decimal points, it will be truncated.

ERROR 1: DATA LIST statement not found

A DATA LIST command is always required.

ERROR 2: MAKETAB statement not found

A MAKETAB command is always required.

ERROR 3: Command NAME not recognized

Where NAME is the first five characters of what DECOMP thought was a command. If all your commands are spelled properly, it could be you have an extra period in somewhere.

ERROR 4 in SET command: Command ignored

This is the only message generated for SET. Processing continues.

ERROR 6: Illegal character found in data.
The character is "X", ASCII decimal Code is YYY at column ZZZ

Where X is an illegal character, YYY is its ASCII code, and ZZZ is the column number of the offending character. All characters other than 0 thru 9 and the decimal point are illegal in data. The character is ignored, and processing continues. Check your data.

ERROR 7: Non-integer found where integer expected in preceding command.
The character is "X", ASCII decimal code is YYY at column ZZZ

Same as ERROR 6, except that the illegal character was found in a command where an integer was expected instead of in the data. ZZZ in this case refers to the column where the offending character would appear is all unnecessary blanks were removed.

ERROR 101 in DATA LIST command: Too many variables listed

A maximum of 30 variables may be specified in a given DATA LIST command.

ERROR 102 in DATA LIST command: Quote mark missing

The 'filename' must be enclosed in single quote.

ERROR 103 in DATA LIST command: Slash not found

Check to see that you are using the correct kind of slash: it should be / not \.

ERROR 104 in DATA LIST command: Command sequence problem

The file declaration must appear before the variable list.

ERROR 105 in DATA LIST command: Too few variables

At least three variables are needed for a DECOMP run.

ERROR 106 in DATA LIST command: Columns illegal

In a column range, the the first column specified must be less than or equal to the second one.

ERROR 107 in DATA LIST command: Cannot read beyond column 200

If your data extends beyond column 200, you must reformat using another program.

ERROR 108 in DATA LIST command: Data file not found

Are the drive and directory correct? They must be specified if the file isn't local.

ERROR 150 in DATA LIST command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what DECOMP interpreted as your variable name; if you left out a column range or a variable name, DECOMP can interpret a column range as a variable name.

ERROR 201 in MAKETAB command: Illegal use of WEIGHT variable

A variable specified in a WEIGHT command may not be used in the subsequent MAKETAB.

ERROR 202 in MAKETAB command: Check punctuation

Probably a typo in the (min,max).

ERROR 203 in MAKETAB command: Too many variables

A maximum of five variables (other than the dependent variable) may be specified.

ERROR 204 in MAKETAB command: independent variable(s) not found

MAKETAB didn't find any variables in your statement.

ERROR 205 in MAKETAB command: Missing DEPENDENT subcommand

A DEPENDENT subcommand is required.

ERROR 206 in MAKETAB command: Missing VARIABLES subcommand

/VARIABLES= is required.

ERROR 207 in MAKETAB command: Subcommands out of sequence

The DEPENDENT= subcommand should appear before the /VARIABLES= subcommand.

ERROR 208 in MAKETAB command: Missing equals sign

You need them for both DEPENDENT= and /VARIABLES=.

ERROR 209 in MAKETAB command: Incomplete statement

Statement should end with close parentheses and period.
Did you leave off the period?

ERROR 210 in MAKETAB command: Invalid minimum or maximum values

Minimum must be less than maximum.

ERROR 211 in MAKETAB command: Too many categories of variables

The product of the range of each variable may not exceed 28,000.

ERROR 250 in MAKETAB command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what DECOMP interpreted as your variable name.

ERROR 251 in MAKETAB command: Minimum and maximum for VARNAME must be specified

Each variable specified after the /VARIABLES= keyword must be supplied with a minimum and maximum value, in the format (min,max).

ERROR 301 in STANDARDIZE command: Illegal use of WEIGHT variable

A variable specified in a WEIGHT command may not be used in a subsequent STANDARDIZE.

ERROR 302 in STANDARDIZE command: Too many BREAKDOWN variables listed

A maximum of five are allowed.

ERROR 303 in STANDARDIZE command: All variables must be from MAKETAB command

You used a variable not described in the preceding MAKETAB command.

ERROR 304 in STANDARDIZE command: Too many CONTROL variables listed

A maximum of four are allowed, except that five are allowed when there are five identical breakdown variables.

ERROR 305 in STANDARDIZE command: Syntax of STANDARD=CATEGORY subcommand

Parentheses OK?

ERROR 306 in STANDARDIZE command: Cannot understand STANDARD subcommand

Probably an invalid keyword; it should be STANDARD=AVERAGE, STANDARD=TOTAL, or STANDARD=CATEGORY(n)

ERROR 307 in STANDARDIZE command: BREAKDOWN variable(s) must be specified

STANDARDIZE found a BREAKDOWN subcommand, but was unable to locate any variables.

ERROR 308 in STANDARDIZE command: Specified STANDARD category out of range

In the STANDARD=CATEGORY(n) subcommand, n must fall between the minimum and maximum specified in the preceding MAKETAB command.

ERROR 309 in STANDARDIZE command: Specified STANDARD category has no cases

DECOMP found no cases in the standard population; they may have been excluded.

ERROR 310 in STANDARDIZE command: All cases excluded from the analysis

This usually indicates a mistake in a SELECT IF or an extremenly small BREAKDOWN category.

ERROR 311 in STANDARDIZE command: Equals sign missing

Syntax must be /BREAKDOWN= and /CONTROL=.

ERROR 312 in STANDARDIZE command: BREAKDOWN subcommand missing

/BREAKDOWN=varname subcommand is required.

ERROR 313 in STANDARDIZE command: CONTROL subcommand missing

/CONTROL=varname subcommand is required.

ERROR 314 in STANDARDIZE command: Option permits only one breakdown category

The STANDARD=CATEGORY(n) option only alows one breakdown variable. If you want more than one, issue another STANDARDIZE command.

ERROR 315 in STANDARDIZE command: CONTROL variable(s) must be specified

STANDARDIZE found a CONTROL subcommand but no CONTROL variables.

ERROR 350 in STANDARDIZE command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be a CONTROL or BREAKDOWN variable.

ERROR 401 in DECOMPOSE command: Illegal use of WEIGHT variable

A variable specified in a WEIGHT command may not be used in a subsequent DECOMPOSE.

ERROR 402 in DECOMPOSE command: All variables must be from MAKETAB command

You used a variable not described in the preceding MAKETAB command.

ERROR 403 in DECOMPOSE command: Too many FACTORS listed

A maximum of four is allowed.

ERROR 404 in DECOMPOSE command: Problem with comparison categories

You must specify two categories in parentheses separated by a colon.

ERROR 405 in DECOMPOSE command: Comparison category out of variable range

Both comparison categories must fall within the range specified in the preceding MAKETAB command.

ERROR 406 in DECOMPOSE command: Too many categories of variables for DECOMP

The product of the range of each variable+1 cannot exceed 28,000.

ERROR 407 in DECOMPOSE command: No cases found in at least one population

Check RECODE and SELECT IF commands and COMPARE subcommand.

ERROR 408 in DECOMPOSE command: Equals sign missing

Syntax is /COMPARE= and /FACTORS=.

ERROR 409 in DECOMPOSE command: FACTOR subcommand not found

/FACTORS=varname is required.

ERROR 410 in DECOMPOSE command: COMPARE subcommand not found

/COMPARE=varname(n1:n2) is required.

ERROR 450 in DECOMPOSE command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be a COMPARE or FACTORS variable.

ERROR 501 in RECODE command: Too many recodes

The total number of input values, defined as individual values or ranges of values, cannot exceed 200 in any pass of the data.

ERROR 502 in RECODE command: Punctuation problem

Check parentheses and equals signs.

ERROR 503 in RECODE command: Range of values invalid

The value preceding THRU must be less than or equal to the value following THRU.

ERROR 504 in RECODE command: ELSE operator illegal

Avoid ELSE by specifying values.

ERROR 505 in RECODE command: Incomplete statement

RECODE must end with a close parentheses and a period. Did you leave off the period of a preceding RECODE?

ERROR 506 in RECODE command: Check syntax

Did you try to use an illegal keyword?

ERROR 550 in RECODE command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be a RECODE variable.

ERROR 601 in SELECT IF command: Operator illegal or missing

Legal operators are GT, LT, GE, LE, EQ and NE.

ERROR 602 in SELECT IF command: Too many SELECT IF statements

A maximum of 50 are allowed in each pass of the data.

ERROR 603 in SELECT IF command: Incomplete statement

SELECT IF must end with a close parentheses and a period. Did you leave off the period of a preceding SELECT IF?

ERROR 650 in SELECT IF command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be a SELECT IF variable.

ERROR 701: Command too long

Maximum of 200 characters allowed, excluding spaces.

ERROR 702 in WEIGHT command

Problem with BY keyword, single quote missing, or problem with the parentheses used to indicate decimal places.

ERROR 703: Array index exceeds 28000

Somehow you managed to trick DECOMP into overrunning the bounds of its main table, without generating a syntax error. This should hopefully never happen.

ERROR 704: Command file not found

Is the path and drive correct? The total length of the drive, path, and DOS filename should not exceed 35 characters. Do not use quote marks.

ERROR 705: Command must begin in column 1

Continuation lines of commands may begin in any column, but the first line of a command must begin in column 1.

ERROR 706: Warning count exceed

This command is ordinarily generated when you have over 25 WARNINGS that a value exceeds the range specified in the MAKETAB command. To fix the problem, you will have to change the range given in your MAKETAB or add a RECODE or SELECT IF statement to get rid of the offending values. The WARNING statement will give you the variable name and case numbers that are causing the problem, so if you cannot figure it out you should examine your data to determine the problem.

ERROR 750 in WEIGHT command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be the WEIGHT variable.

ERROR 801 in COMBINE command: Punctuation problem

Check the sequence of parentheses and commas.

ERROR 802 in COMBINE command: Too many variables

A maximum of four variables may be combined in one statement. If you need more, break it into two statements (COMBINE your previously combined variable).

ERROR 803 in COMBINE command: Invalid minimum or maximum values

Minimum must be less than maximum.

ERROR 804 in COMBINE command: Incomplete statement

Statement must end with a close parentheses followed by a period.

ERROR 805 in COMBINE command: You must have two or more variables

Only one variable to be combined was identified.

ERROR 806 in COMBINE command: Too many categories of variables

The product of the ranges of your variables to be combined must not exceed 28,000. If it's even close, you probably don't want to do it.

ERROR 850 in COMBINE command: Variable VARNAME not found

Where VARNAME is the first 10 characters of what the program interpreted to be the COMBINE variable.