Creating Inferred Data Using SAS®

Tim Walters, *InfoTech Marketing*, Littleton, CO

Abstract

Because of the costs and difficulty of data collection, companies often do not collect all the data an analyst needs. Base SAS provides a number of tools to create inferences, or "inferred data", from data that is commonly collected. This paper describes various data that can be inferred, procedures for standardizing data, three programming techniques to generate inferred data, and limitations on creating and using inferred data.

Introduction

Often, users need data that could be collected but isn't because of costs, difficulty, or completeness. For example, an analyst in a company marketing to businesses may want to know the department (CEO, Sales, Marketing, Engineering, etc.) who is buying their goods or services. This data is seldom collected, or may be partially collected by having the buyers return questionnaires. Base SAS contains a number of tools allowing you to create such data when it is missing, which I call "inferred data". This paper describes three basic ways to create inferred data using SAS.

Definition of Inferred Data

Inferred data is that which could be collected but isn't. Inferred data is data, pure and simple. It is differentiated from information, or processed data, in that inferred data could be directly collected but isn't. For example, classifying a customer based on size (small, medium, or large) from other data within their own database is inferred data. Overlaying data from as outside company that has company size is not inferred, because the data has been directly collected. Classifying a customer based on usage of a product (light, medium, and heavy) is not inferred data, because this data couldn't be collected from the customer without processing their purchase patterns. This distinction is somewhat arbitrary and difficult to describe, but should become clearer throughout the examples presented.

Types of Data That Can Be Inferred

Many types of data can be inferred using Base SAS. The author's background is primarily in marketing and finance, so most of the examples are from these areas. In marketing, much customer information can be inferred. For instance, a purchaser's department and level within the company can be inferred from their title.

Data can be inferred in at least three ways. First, you can decompose a variable that is collected. For instance, by analyzing the title of the buyer, you can infer their department. Second, a comparison of two data fields with a decision rule can also be used. If the customer's phone number is the same as their fax number, you can infer that the company is small, for larger companies have phone numbers different from their fax. Third, you can aggregate a variable by customer to obtain useful inferences. For costing purposes, a truckline may want to know what shipments were picked-up at a customer stop for the day. Because the cost of obtaining, entering, and verifying the data may be prohibitive, the company may decide to infer what shipments are picked-up at each stop from data on the number of shipments picked-up at each customer by day. Although this data might be considered information, as opposed to inferred data, I still regard it as inferred data because it could be obtained from the customer if so desired.

Data Standardization

One of the first steps in creating inferred data is data standardization. Standardization can eliminate case sensitivity problems, eliminate unwanted data that confuses analysis, and create variables with a consistent format. SAS provides a number of tools to accomplish this.

Case Sensitivity

One of the first steps of data standardization in dealing with text variables is to convert them to a common case. UPPER CASE is the traditional case used. SAS provides the UPCASE function that converts all characters in a text variable to upper case. For instance, if you are analyzing the title of your customers, the following SAS code creates a new variable (TITLE) of all UPPER CASE text from the original title variable (BTITLE):

```
TITLE=UPCASE(BTITLE);
```

Elimination of Unwanted Data

After a variable is standardized for case, it is often important to eliminate unwanted characters. For instance, customer titles are often inconsistently entered into the customer database. Spaces are often applied inconsistently (C.E.O. and C. E. O.). Abbreviations may also be used with periods applied differently (CEO and C.E.O.). Dashes are sometimes used between the rank and department, while other titles use the word "of" (Director—Marketing and Director of Marketing).

One of the most useful SAS functions for eliminating unwanted characters is the COMPRESS function. Used without a second argument, COMPRESS eliminates all blank spaces. Using the previously created TITLE variable, this code eliminates the blanks in TITLE in the new variable CTITLE:

```
CTITLE=COMPRESS(TITLE);
```

The COMPRESS function can also be used to eliminate specific unwanted characters by listing them in the second argument. This code eliminates periods, dashes, percent signs, and other miscellaneous characters that may have been entered into the original title field in the new variable DTITLE:

```
DTITLE=COMPRESS(CTITLE,',-%$(&)');
```

Elimination of a specific word is slightly more complex but can be readily accomplished. Let's say you want to eliminate the word "and" from the DTITLE variable created above. The first step is to use the INDEX function to find the first character of the word "and" in the title and assign it to a variable (A). Because the UPCASE function has already been applied, AND must be specified as uppercase. If AND is not found, the INDEX function sets the A variable to 0. If AND is found, A equals the number of spaces to the first occurrence of "A" in "AND". If A is not equal to 0, it means that AND was found, and the SUBSTRING and TRIM functions, along with the concatenation operator, are used to reconstruct the title without the AND string. The following code illustrates this:

```
A=INDEX(DTITILE,"AND");

IF A NE 0 THEN ETITLE =
TRIM(SUBSTR(DTITLE,1, A-1)) ||
TRIM(SUBSTR(DTITLE,A+3));

ELSE ETITLE=DTITLE;
```

An example of using these statements clarifies their effect. Let's assume your data set includes the title variable in a customer master file as shown as the BTITLE variable in Figure 1. Applying the statements shown above for case sensitivity and elimination of unwanted data yields the ETITLE variable also shown in Figure 1.

Case Sensitivity/Elimination of Unwanted Data

OBS	BTITLE -- Original Data	ETITLE -- Standardized Data
1	Admin. Asst.	ADMINASST
2	CEO	CEO
3	C.E.O.	CEO
4	CEO & President	CEOPRESIDENT
5	CEO and President	CEOPRESIDENT
6	Vice President -- Western Region	VICEPRESIDENTWESTERNREGION
7	CEO and President%	CEOPRESIDENT
8	V.P. of Sales	VPOFSALES
9	Dir. Multilevel Marketing	DIRMULTILEVELMARKETING
10	Controller	CONTROLLER
11	CFO & Controller	CFOCONTROLLER

Figure 1. Case Sensitivity/Elimination of Unwanted Data Output

Consistent Formatting

In certain cases, data manipulation may be necessary to create consistently formatted variables. For instance, if telephone number is entered as a text variable, some phone numbers could be entered as just 10 consecutive numbers (8885551212), some may be entered with spaces between them (888 555 1212), some may be missing the area code for where you are located (5551212), etc. (Of course, this assumes most or all of the telephone numbers are from the U.S. or Canada. International telephone numbers create additional problems.) For telemarketer ease of use, you may want to standardize the phone numbers in a report with dashes between the area code and exchange and exchange and number (888-555-1212). SAS offers a number of text manipulation tools to assist in this process.

Probably the easiest way to begin is by using the COMPRESS function to eliminate spaces and extraneous characters. As shown above, these two compress statements accomplish this:

```
APHONE=COMPRESS(PHONE);

BPHONE=COMPRESS(APHONE,",-%$(&)");
```

To determine the presence of an area code, utilize the LENGTH function. The LENGTH function returns the number of text characters in a string. For properly entered phone numbers, the value of the variable created with the LENGTH function should either be seven (no area code) or ten (area code present). A relevant example of the LENGTH statement is:

```
VARLEN=LENGTH(BPHONE);
```

Once you determine the presence or absence of the area code, you either need to transfer it to the new variable being created, or insert the area code of where you are

located. The SUBSTR function and concatenation operator build a new consistently formatted variable from what already exists:

```
IF VARLEN=10 THEN NEWPHONE =
SUBSTR(BPHONE,1,3)||"-"||SUBSTR(BPHONE,4,3)
||"-"||SUBSTR(BPHONE,7,4);

IF VARLEN=7 THEN NEWPHONE="888-
"||SUBSTR(BPHONE,1,3)||"-"||SUBSTR(BPHONE,4,4);
```

Using the above statements transform the raw data shown in Figure 2 as PHONE to the standardized NEWPHONE variable. Note that the last observation has not been transformed because its length (the VARLEN) variable is neither 7 nor 10. You may want to print all observations that are not 7 or 10 for manual review and updating before the final report is produced.

Consistent Formatting

OBS	PHONE	VARLEN	NEWPHONE
1	8885551212	10	888-555-1212
2	888 555 1212	10	888-555-1212
3	888-555-1212	10	888-555-1212
4	(888) 555-1212	10	888-555-1212
5	5551212	7	888-555-1212
6	555-1212	7	888-555-1212
7	55512121	8	

Figure 2. Consistent Formatting Output

Techniques for Creating Inferred Data

One Variable Decomposition

Once the data is standardized, it can then be analyzed. One form of analysis is one variable decomposition. By examining one variable, various inferences can be made. For instance, if you have a telephone number that ends in "00", it probably denotes a large company. Similarly, if digits 3 and 4 of a ZIP code are "99", you can infer that this is a large company that has its own ZIP code. Figure 3 contains other inferences that can be made based upon examining one variable.

Variable Contents	**Inference**
Phone Number – Last two digits ="00"	Large Company
Phone Number – Extension Shown	
ZIP Code – Digits 3 & 4 = "99"	Large Company
Title – Detailed Analysis	Department and Rank/Job Function
First Name – Detailed Analysis	Gender
Last Name – Detailed Analysis	Ethnicity
Company Name – Detailed Analysis	Industry
Company Name – "Inc."	Corporation
Company Name – "LLC"	Limited Liability Corporation
Company Address – Suite Number Present	Office Location
E-mail Address at Large On-line Services Provider (AOL, CompuServe, Prodigy)	Small Company
E-mail Address Contains .GOV	Governmental Body
E-Mail Address Contains .ORG	Association
E-Mail Address Contains .NET	Large Company

Figure 3. Inferences That Can Be Made from One Variable Decomposition

Some inferences can be more strongly made than others can. For instance, the limitation of using company name to infer industry is that you do not know where the company is in the distribution chain. A company named "Cement Unlimited" probably has something to do with cement, but it cannot be determined if the company is a manufacturer, wholesaler, or retailer. Gender analysis based on first name can also be confusing with names like "Francis", "Pat", and "Mickey". In some cases, the inference may not be applicable to your industry. In many cases, ethnicity based upon last name is irrelevant in marketing.

Many of the SAS tools used for standardizing data apply as well in performing one variable decomposition. For instance, the INDEX function can find "00" in the phone number. Another alternative is to create a consistently formatted telephone number (see above) and employ the SUBSTR function to take digits 11 and 12 and compare them to "00".

For the variables shown above requiring detailed analysis, you can look for a certain string within the variable and assign it to a category using "IF/THEN" or "SELECT" statements. In the following example, a person's title will be decomposed to infer their rank (C.E.O., Director, Manager, etc.) or job function (account executive, attorney, etc.) The INDEX function (see Figure 4) is used to locate a certain string within the already standardized title (which, continuing from above, is the ETITLE variable). If the INDEX function does not find the string, it returns a 0. If the INDEX function finds the string within the ETITLE variable, it returns the first location of the string. If the result of the INDEX function, therefore, is not 0, it signifies that the string has been found. Using the IF/THEN statements shown in Figure 4, if the INDEX function finds the string, it then gets assigned to the new variable, NTITLE, which is used to categorize the title as to rank or job function.

```
LENGTH NTITLE $16;
  IF INDEX(ETITLE,'SECRETARY') NE 0 THEN NTITLE='ADMIN. SUPPORT';
  IF INDEX(ETITLE,'RECEPT'   ) NE 0 THEN NTITLE='ADMIN. SUPPORT';
  IF INDEX(ETITLE,'EXECSEC'  ) NE 0 THEN NTITLE='ADMIN. SUPPORT';
  IF INDEX(ETITLE,'EDITOR'   ) NE 0 THEN NTITLE='EDITOR        ';
  IF INDEX(ETITLE,'SALESREP')  NE 0 THEN NTITLE='A.E.          ';
  IF INDEX(ETITLE,'ACCOUNTEX') NE 0 THEN NTITLE='A.E.          ';
  IF INDEX(ETITLE,'ACCTEXEC' ) NE 0 THEN NTITLE='A.E.          ';
  IF INDEX(ETITLE,'ENGINEER')  NE 0 THEN NTITLE='ENGINEER      ';
  IF INDEX(ETITLE,'ENGR'     ) NE 0 THEN NTITLE='ENGINEER      ';
  IF INDEX(ETITLE,'APPLENG' )  NE 0 THEN NTITLE='ENGINEER      ';
  IF INDEX(ETITLE,'ATTORN')    NE 0 THEN NTITLE='ATTORNEY      ';
  IF INDEX(ETITLE,'ATTY'  )    NE 0 THEN NTITLE='ATTORNEY      ';
  IF INDEX(ETITLE,'QATECH' )   NE 0 THEN NTITLE='ANALYST/SPEC. ';
  IF INDEX(ETITLE,'LIBRARI')   NE 0 THEN NTITLE='ANALYST/SPEC. ';
  IF INDEX(ETITLE,'CHMST'  )   NE 0 THEN NTITLE='ANALYST/SPEC. ';
  IF INDEX(ETITLE,'COMMANAL')  NE 0 THEN NTITLE='ANALYST/SPEC. ';
  IF INDEX(ETITLE,'ACCT'     ) NE 0 THEN NTITLE='ACCOUNTANT    ';
  IF INDEX(ETITLE,'CNTRLR'   ) NE 0 THEN NTITLE='ACCOUNTANT    ';
  IF INDEX(ETITLE,'AUDITOR'  ) NE 0 THEN NTITLE='ACCOUNTANT    ';
  IF INDEX(ETITLE,'CPA'      ) NE 0 THEN NTITLE='ACCOUNTANT    ';
  IF INDEX(ETITLE,'SUPER')     NE 0 THEN NTITLE='SUPERVISOR    ';
  IF INDEX(ETITLE,'SUPR' )     NE 0 THEN NTITLE='SUPERVISOR    ';
  IF INDEX(ETITLE,'MGR')       NE 0 THEN NTITLE='MANAGER       ';
  IF INDEX(ETITLE,'MANAGER')   NE 0 THEN NTITLE='MANAGER       ';
  IF INDEX(ETITLE,'GENMGNT')   NE 0 THEN NTITLE='MANAGER       ';
  IF INDEX(ETITLE,'GM'     )   NE 0 THEN NTITLE='MANAGER       ';
  IF INDEX(ETITLE,'DIRECTO')   NE 0 THEN NTITLE='DIRECTOR      ';
  IF INDEX(ETITLE,'EXECDIR')   NE 0 THEN NTITLE='DIRECTOR      ';
  IF INDEX(ETITLE,'SRDIR'  )   NE 0 THEN NTITLE='DIRECTOR      ';
  IF INDEX(ETITLE,'DIR ' )     NE 0 THEN NTITLE='DIRECTOR      ';
  IF INDEX(ETITLE,'VICEPR')    NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'OFFICER')   NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'VP')        NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'CHIEFFIN')  NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'CHIEFINFO') NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'CFO'     )  NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'CIO'     )  NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF INDEX(ETITLE,'PRINCIP')   NE 0 THEN NTITLE='PRINCIPAL     ';
  IF INDEX(ETITLE,'CEO')       NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'DEAN ')     NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'PUBLISHER') NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'PRESID')    NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'PRES ' )    NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'CHAIRMAN')  NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'CHIEFEXE')  NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'PASTOR'  )  NE 0 THEN NTITLE='C.E.O.       ';
  IF INDEX(ETITLE,'OWNR' )     NE 0 THEN NTITLE='OWNER         ';
  IF INDEX(ETITLE,'OWNER')     NE 0 THEN NTITLE='OWNER         ';
  IF INDEX(ETITLE,'ONWER')     NE 0 THEN NTITLE='OWNER         ';
  IF INDEX(ETITLE,'VICEPRES') NE 0 THEN NTITLE='SENIOR MGMT.  ';
  IF NTITLE =' ' THEN NTITLE='OTHER';
```

Figure 4. Statements to Infer Job Function/Rank

Because the program uses a series of IF/THEN statements, the position of the statements within the program is important. The program places the title into a category and then may later change the category based on program statements located further down in the code. For example, a title of "Vice President" originally is assigned to the "President" category, but is later re-classified as "Senior Mgmt."

Using the type of program shown in Figure 4 on my own database of contacts and prospects, combined with the PROC FREQ procedure, produces the results shown in Figure 5. Of the 650 contacts with a title, only 5.2%, or 34, are assigned to the "Other" category of those that cannot be identified. The program, therefore, successfully assigns about 95% of the titles to a category.

Title Analysis Results

NTITLE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
A.E.	10	1.5	10	1.5
ACCOUNTANT	1	0.2	11	1.7
ADMIN. SUPPORT	12	1.8	23	3.5
ANALYST/SPEC.	10	1.5	33	5.1
ATTORNEY	1	0.2	34	5.2
C.E.O.	200	30.8	234	36.0
CONSULTANT	7	1.1	241	37.1
DIRECTOR	80	12.3	321	49.4
EDITOR	18	2.8	339	52.2
ENGINEER	2	0.3	341	52.5
MANAGER	80	12.3	421	64.8
OTHER	34	5.2	455	70.0
OWNER	58	8.9	513	78.9
PARTNER	7	1.1	520	80.0
PRINCIPAL	2	0.3	522	80.3
SENIOR MGMT.	125	19.2	647	99.5
SUPERVISOR	3	0.5	650	100.0

Figure 5. Title Analysis Results

Just because the program assigns titles to 95% of the contacts does not necessarily mean that the program assigns the title correctly. Visual examination of the results should be performed to gain a level of confidence that the tiles are in fact being correctly inferred.

Two Variable Comparison

Another technique to infer data is two variable comparison. For instance, if a company's phone number equals its fax number, it is probably a small business. If the last name of the contact (Brown) is contained within the name of the business (Brown's Real Estate), it is also likely to be a small business. The table in Figure 6 shows other possible inferences to draw by comparing two variables.

Variable One	Variable Two	Inference
Phone Number	Fax Number	If Equal, Small Company
Contact Name	Company Name	If Contained Within, Small Company
Company Name	Street Address	If Contained Within, Large Company (except common street names like "Broadway Diner")

Figure 6. Two Variable Comparison Inferences

In addition to these generic comparisons, there may be other comparisons dependent on the type of business and data you want. For instance, in a telecommunications company, you may be interested in knowing where someone placed a call. If the phone number of the call detail record matches the customer's contact phone number, they probably made in from their own phone. If only the area code and exchange of the call detail record matches the area code and exchange of the customer's master record, they probably made the call from another phone near their own phone. If only the area code matches, they made the call from a different phone within their own area code. Each type of business probably has

other examples of where two variable matching can be used to infer customer behavior.

The methods for performing two variable comparisons are straightforward. To compare phone number and fax number, simply use the "=" operator and assign the results to a variable, as shown below:

```
  IF PHONENO=FAXNO THEN COMPSIZE =
"SMALL";
```

To find a name within another character variable, you can use the index function as shown in Figure 4.

One Variable Aggregation

Another technique for inferring data is aggregating one variable. For example, in trucking, it is important for costing purposes to know how much freight is picked-up from the customer each time the driver stops at the customer's premises. This data could be collected and put into a database with cross-references to the actual shipment records, but often is not because of the additional cost of collecting the data. The amount picked-up, therefore, is often inferred from the shipment information.

SAS provides a number of tools for data aggregation. If you do not have a lot of records or classes and there are no constraints on your data, the simplest tool is PROC SUMMARY. In the trucking example, you can summarize the weight and number of shipments picked up by day by using the class variables of customer and shipment day. The following code shows how:

```
PROC SUMMARY NWAY;
  CLASS CUSTOMER SHIPDATE;
  VAR SHIPWGT SHIPMENT;
  OUTPUT OUT=SUMM SUM=;
```

This code creates a total record for each customer for each day (by using the SUM= option on the OUTPUT statement) of the weight and number of shipments shipped in work file SUMM.

If you have a lot of classes or want to speed up your program, a more efficient way of aggregating is to sort the data by the class variables you would otherwise use and sum them in a DATA step. The following code shows this alternative to the PROC SUMMARY.

```
PROC SORT;
  BY CUSTOMER SHIPDATE;

DATA SUMM;
  SET INP;
  BY CUSTOMER SHIPDATE;
  IF FIRST.SHIPDATE THEN DO;
    TSHIPWGT=0;
    TSHIP=0;
  END;
  TSHIPWGT+SHIPWGT;
  TSHIP+SHIPMENT;
  IF LAST.SHIPDATE;
```

After sorting the data, the DATA step reads in the file with the BY variables in the sort. When it comes to the first record for a day for each customer, the two variables that will contain totals are initialized to 0. As each observation is read, the weight and shipment count (usually 1) are added to the total variables. The last statement only outputs the last observation of each day for each customer. This observation should have the total weight and shipments for that day. The output is exactly the same as the output from the PROC SUMMARY procedure except for different variable names.

Once the data has been aggregated, the totals need assignment to each individual observation. This can be accomplished by merging the summary data file with the original detail data file, as shown below.

```
DATA FINAL;
   MERGE INP SUMM;
   BY CUSTOMER SHIPDATE;
```

Each individual observation will then include not only its weight and shipment count, but now also has the total weight and shipment count for that day.

The above two techniques implicitly assume that there are no constraints on the data. In fact, for most types of shipments, the maximum a truck can pick up at one time is 15,000 pounds, so a truck capacity constraint may be introduced. If the customer has a lot of shipments in a given day totaling over 15,000 pounds, it may be assumed that multiple pick-ups are required that day. This scenario can be accommodated by using the DATA step as above but including a constraint variable with a RETAIN statement.

A further assumption is made that the largest weight shipments are picked-up first. This is an arbitrary assumption made in order to sort and aggregate the data (note the DESCENDING SHIPWGT portion of the BY statement in the following code).

```
PROC SORT DATA=INP;
  BY CUSTOMER SHIPDATE DESCENDING;
RUN;
DATA SUMM;
  SET INP;
  BY CUSTOMER SHIPDATE DESCENDING;
  IF FIRST.SHIPDATE THEN DO;
    TSHIPWGT = 0;
    TSHIP = 0;
    STOPPDAY = 1;
  END;
  RETAIN TSHIPWGT TSHIP STOPPDAY;
  TSHIPWGT + SHIPWGT;
  TSHIP + SHIPMENT;
  IF TSHIPWGT GT 15000 THEN DO;
   TSHIPWGT = SHIPWGT;
    TSHIP = SHIPMENT;
    STOPPDAY = STOPPDAY + 1;
  END;
RUN;
PROC SORT DATA=SUMM;
  BY CUSTOMER SHIPDATE STOPPDAY DESCENDING TSHIP;
RUN;
DATA FINAL;
  SET SUMM;
  BY CUSTOMER SHIPDATE STOPPDAY;
  IF FIRST.SHIPDATE OR FIRST.STOPPDAY THEN DO;
    NSHIPWGT = TSHIPWGT;
    NSHIP = TSHIP;
  END;
  RETAIN NSHIPWGT NSHIP;
  DROP TSHIPWGT TSHIP;
RUN;
```

Once the data is sorted and the DATA step begins executing, a constraint variable called TSHIPWGT is initialized and set to 0 before each day's shipments by each customer is began to be read. Variable TSHIP is used to count the shipments and is initialized to 0 for each day. The number of stops for each day will be counted by the STOPPDAY variable, which is initialized to 1. The RETAIN statement retains the value of each of these variables from one observation to the next. As each observation is read, the weight and number of shipments are added to TSHIPWGT and TSHIP, respectively.

If the TSHIPWGT exceeds 15000, the program executes a DO loop that re-initializes the three retained variables. TSHIPWGT and TSHIP are re-initialized to the weight and shipment count of the observation that would put it over the 15,000 pound constraint. The STOPPDAY variable is incremented by 1 to show that another pick-up is required.

In order to place the weight and shipment of each stop on each individual observation comprising the stop, the data is sorted by customer, the date, the stop number within that date, and the descending shipment count. The FINAL data set is then created. If the observation is the first observation for that date or for that stop, the program initializes variables NSHIPWGT and NSHIP with the total weight and shipment for that stop. The NSHIPWGT and NSHIP variables are retained in order to assign them to all observations in the stop.

Limitations

This paper has presented techniques for creating inferred data. As such, inferred data is only educated estimates and not as accurate as collecting the data directly. People looking for exact data will be disappointed with inferred data and probably attack its credibility.

Additionally, some data is very difficult to even infer and may lead to incorrect inferences. The company must weigh the risks of incorrect inferences versus no inferences at all. If the cost of incorrect inferences is greater than none at all, then the company should not create inferences for that data.

Inferred data is sometimes only a partial answer. For instance, if a company's phone number equals its fax number, you may be able to infer that the business is small, but there are also many small businesses whose phone number is not equal to their fax number.

Summary

Inferred data can be used to "fill-in-the blanks" on observations lacking data you collect, or to add completely new variables to the data you already have. SAS gives you a number of ways to create inferred data, of which this paper enumerates three. Is it worth it? While not being entirely accurate, the pay-off from creating inferred data is the better decision-making that it enables. This is sometimes referred to as the value of imperfect information. In most cases, this value will dramatically exceed the programming costs required to extract it, so go for it.

Acknowledgments

Contact Information

Tim Walters
InfoTech Marketing
9350 W. Cross Drive, Suite 203
Littleton, CO 80123
800-506-0252 (Voice)
303-727-4690 (Fax)
InfoTecMkt@aol.com (E-Mail)

A SAS Institute Quality Partner