PhUSE 2017

Paper DH08

Data Cleaning Technique for Business Analytics. What Stats Can Do for DM?

Adam Korczyński, QuintilesIMS, Warsaw, Poland

ABSTRACT

Imagine that you receive your revenue from goldsmith on a monthly basis but it comes in irregular shapes, has missing parts etc. as if it was a piece of gold found in a remote Neolithic site. This would certainly raise questions like: How to estimate the founds I have? Can I make proper financial decisions? On the other hand you know what you want and in order to get closer to the ideal world and meet your expectations you need to inform the goldsmith on what needs to be done prior to the transfer. That will save the time and effort on both ends. That is the essence of the approach to data cleaning suggested in the paper. In this case what you get is not directly gold but data and what you need is knowledge which relies on data quality.

In practice data quality is driven by the cooperation of the data provider and the analyst. The aim of the paper is to present a data cleaning technique, which serves for parties involved in data collection as well as for the final data user. The analyst is able to provide an overview of the final requirements and to develop a standardized tool for data checks which can be used by data management to minimize the effort related to data queries. This input allows identification of clearly erroneous data but also data values that appear to be unusual. The key is to make a benefit of the knowledge possessed by the analyst at the initial stage of data processing. The paper enumerates the most common data issues and outlines a data cleaning technique implemented in a macro program written in SAS® v9.2. The technique is exemplified in a clinical trial data.

INTRODUCTION

Data is an asset for the company (Laursen & Thorlund, 2010, pp. 179-80). This specific asset requires to be managed properly. Quality of data affects work efficiency, which impacts the competitiveness of a company, its services and profits. High quality data ensures on-time reporting and accurate organization of work. Whenever a company needs to deal with large amounts of data or with complicated process of data collection the data quality relies on the cooperation between Analytical team (Stats) and Data Management (DM). The better the quality of the data received by the analyst the more efficient and shorter the work of the analysts is.

In practice the processes of work of Stats and DM are different, mostly due to having distinct perspective on the data. The role of DM is to collect, verify, query and transfer data whereas Stats is responsible for final reports based on which the business decisions are made. Both teams have their own areas of responsibility but there are some common activities that can be pointed out (see Figure 1). The question is at which point can both departments meet? In consequence what could be done to make the cooperation efficient?

The aim of the paper is to suggest a data cleaning technique, which serves for both data manager and data analyst. The crucial element is that this technique requires the input of both sides: initial user (DM) and final user (Stats). An analyst is able to provide a wide overview of the final requirements i.e. to say what's not possible; to indicate erroneous data; to point out what's unusual; to identify deviations in the process under analysis (Isson & Harriott, 2013, pp. 72-3); and highlight what's equivocal (Reis Pinheiro & McNeill, 2014, pp. 119-23). Most likely the analytical team will be able to provide the tool for these checks. DM is then responsible for the initiation of the checks, as well as the maintenance and upgrades to the tool. The tool does work on the intersection of data management and analytical tasks as it is shown on Figure 1. Its aim is to make progress towards the data quality through data validation process. The general idea is to make a benefit of the knowledge possessed by the analyst at the stage of the initial data processing. Such an input, implemented by a check carried out at the initial stage limits the time needed to query data later on. It is beneficial for DM as it brings a reduction in number of data issues and thus in result the analyst has more time for the analysis itself.

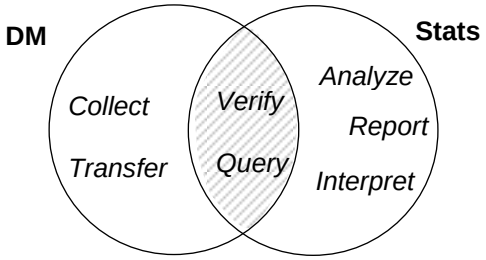


Figure 1 The tasks of DM and Stats teams
Source: Own elaboration.

The paper is composed of three sections. The first section describes the idea of the data cleaning technique with its programming details. The second section outlines most common data issues with suggestions for the potential data checks. The third part exemplifies the depicted technique on a clinical trial data. A macro program written in SAS v9.2 by which the technique can be implemented can be found in the Appendix.

DATA CLEANING SCHEME FOR STRUCTURED DATABASES

In the simplest example data cleaning and analysis is carried out on a single structured dataset. In practice the analyst usually has access to the database made of some number of datasets with different data. In this section we refer to the structured database created according to the relational model as opposed to unstructured data which is not organized into tables with specific indexing rules. An example of non-structured data would be a text from a Web forum, see also (Ohlhorst, 2013, pp. 5-6; Isson & Harriott, 2013, pp. 359-76).

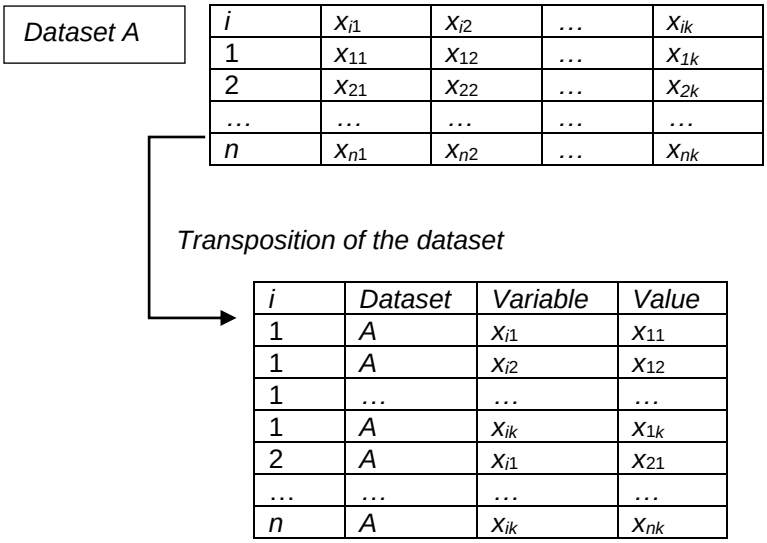
In order to discuss the data processing steps let us first consider the very basic structure of a dataset. Single dataset can contain unique information for a particular object (like identification number and basic characteristics) or multiple information per object (like list of characteristics varying within an object). A special and important example of a dataset with multiple information per object would be a time series.

As there are differences between datasets with regards to data type and structure, the first step in the data cleaning technique consists of unification of the data. The issues with data can occur at any place within the dataset. A good technique should allow identification of an issue wherever it is. In relational database environment – to which we refer in this paper – any place means any row and any column. The proposed unification is to create a single dataset with one column representing all the entries from the input dataset. By having all the entries in one column we already make the checks on data straightforward. The only further step needed is to filter or scan the data in that column to look for any problematic entries.

Because of the specificity of a relational database we assume that each dataset that we want to check will have an identification column (*i*). This column enables us to locate the issues across datasets. The operation to perform is to transpose that dataset in order to get a new row for each value within each single observation in the input dataset. To explain that further let us consider the following notation.

Let us assume that in dataset *A* we have identification variable *i* - where $i = 1, 2, \ldots, n$ - and *k* explanatory variables $x_{i1}, x_{i2}, \ldots, x_{ik}$. The transposition of the dataset *A* would mean to create a column based on each row of values for variables $x_1, x_2, \ldots, x_k$ for each unique *i* as it is shown on Figure 2. Taking the first object ($i = 1$) as an example we see that the initial values $x_{11}, x_{12}, \ldots, x_{1k}$ are stored under a single variable called *Value* in the output dataset. The same operation is done to all of the other objects and as a result we get a transposed dataset. As we can see this dataset contains not only the values of the variables in the initial dataset but also some other descriptive characteristics enabling us to identify each entry. A minimal set of descriptive features would consist of the input dataset name and the name of a variable the value of which is stored in *Value*.

It is worth mentioning that the transposition would rather involve the usage of metadata. By using metadata we can obtain a list of variables in each dataset within a directory and then we can use this information in order to identify the variables to be transposed. It is very handy to use metadata for that purpose because it relieves us from the necessity of any checks against the datasets specifications at that stage. This is in line with the general idea which is to check the data as is without any prior knowledge on the content.



i	x_{i1}	x_{i2}	…	x_{ik}
1	x_{11}	x_{12}	…	x_{1k}
2	x_{21}	x_{22}	…	x_{2k}
…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{nk}

i	Dataset	Variable	Value
1	A	x_{i1}	x_{11}
1	A	x_{i2}	x_{12}
1	…	…	…
1	A	x_{ik}	x_{1k}
2	A	x_{i1}	x_{21}
…	…	…	…
n	A	x_{ik}	x_{nk}

Figure 2 Transposition of a dataset
Source: Own elaboration.

Let us further assume that we have I datasets A_1, A_2, …, A_I. The second and third steps of the data cleaning technique are respectively to transpose each dataset in a directory and combine these datasets afterwards to get one final dataset including all entries from the database[1], as shown on Figure 3.

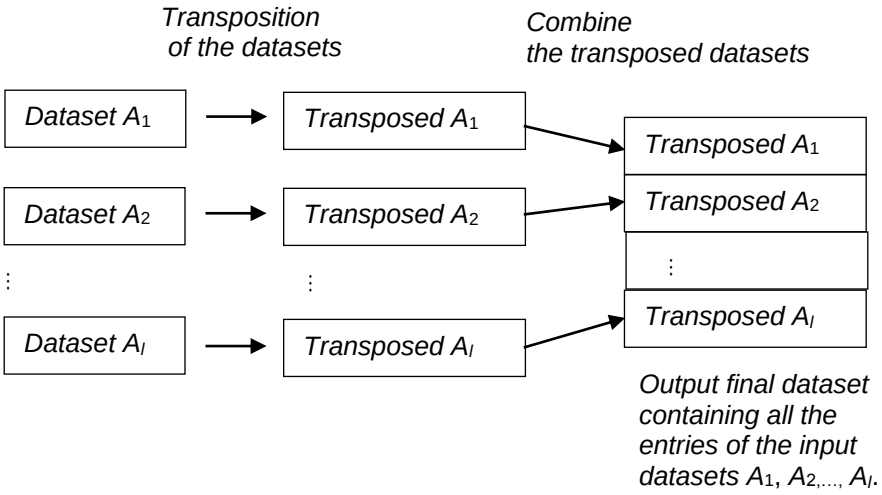


Figure 3 Combining data into the final dataset
Source: Own elaboration.

An example of a final transposed dataset created using *TransposeAllLib* macro program (see Appendix A) is shown on Figure 4. In clinical practice the dataset will contain patient ID, study code and line - a variable representing multiple observations within patient. The key metadata are source dataset name (MemName), library (LibName) and characteristics of the variables in the source dataset: variable name (ParamCd), label (ParamLb), format (ParamFt) and type – character or numeric (ParamTp). The actual values are in a set of variables in order to be able to differentiate between character and numeric outcomes and to take the formatting into consideration. The variables with the actual values are as follows: ParamVlC – original value represented as a character, ParamVlCD – formatted value represented as a character, ParamVl – original numeric result or 1 (for character values) and ParamVlO - original numeric result or missing (for character values).

patient	study	line	memname	libname	ParamCd	ParamLb	ParamFt	ParamTp	ParamVlC	ParamVlCD	ParamVl	ParamVlO
290	999A	2	AEA	RAW	A_V	V. Outcome - a	BEST12.	N	2	2	2	2
290	999A	2	AEA	RAW	A_EVENT	Not on CRF (text code) - a	$19.	C	HEADACHE	HEADACHE	1	.

Figure 4 Example of a final transposed dataset
Source: Own elaboration based on database from the study NIDA-CPU-0016 (2014).

Once the final dataset is available the last step is to apply any potential filtering or scanning for data issues on the *Value* variable in the final output dataset. This is the stage where the input of the analyst is vital. Only by knowing the final purpose of the data we can imagine a full set of checks required in order to clean the initial entries. We shall explain some of the plausible checks applicable to any area of business analytics. Two checks for common issues are exemplified in the application section of this paper.

DATA TYPES AND CHECKS

In order to outline any potential data issues let us first consider the types of data. Statistical theory traditionally uses the scale to distinguish the types of data. The main types of data are (Hoerl & Snee, 2012, pp. 363-6): nominal, ordinal, integer, interval and ratio. These scales are growing in number of operations that can be performed based on each scale.

Nominal data enables only to say whether the values are the same or different. Sex would be an example of a variable measured on nominal scale.

Ordinal scale enables additionally to express preferences over the values i.e. we can say that one value is greater/lower (better/worse) than another based on some logic order. That would be the case of education level. The ordinal scale does not provide the magnitude of the distance between particular categories – this is the additional feature of the interval scale.

Interval data enables to say not only that one value is greater/lower than the other one but also to say how much greater/lower it is. This is the feature of the continuous variables. An example would be temperature data in Celsius degrees or calendar data.

Integer scale, capturing counts data, is a very special scale, which is in between ordinal, and interval scale. On one hand it allows operations expressing preferences as well as it measures the distance e.g. 4 claim counts are greater than 1 by 3 where 3 expresses the distance between the two values. However it is not true continuous scale.

Interval scale does not define the absolute zero. Going back to the temperature data measured in Celsius degrees, zero on that scale is just a convention. Then we cannot actually say how many times is -10ºC lower than 20ºC. Similarly it is not possible to answer the question about how many times 10 March 2016 is greater than 10 February 2016.

We say that the measurements having an absolute zero are on ratio scale. This scale is relevant for variables like weight and income. We can say that 20 kg is two times as heavy as 10 kg. As opposed to the calendar date example, time can be considered as measured on ratio scale, e.g. we can say that a three months therapy is three times greater than one-month therapy.

The traditional scales can be translated into three main data types, which are important from programming perspective: characters, numbers and a special type of number, which is date and time. Characters are nominal and ordinal data. Numeric data captures integer, interval and ratio data. For the purpose of the data checks date and time are considered as interval data most frequently. The issues with date and time are mostly related to the calendar days.

The types of data implicate special issues to consider, see also (Winkler, 2004, p. 533; Isson & Harriott, 2013, p. 73):

- Character data: hidden characters, not standardized entries and spelling mistakes;
- Numeric data: outliers, unusual distribution of numbers;
- Date and time: partial dates/times, not-possible dates/times, unusual dates/times, and equivocal dates/times.

We can also indicate some issues applicable to any data type, e.g. missing data and duplicates. Let us revise the above issues and indicate their potential implication for data-driven business decisions.

Textual data entered manually into the system is prone to erroneous entries like hidden or mistaken characters. Hidden characters e.g. line feed, lead to a situation when two “visually” identical strings are differing which causes problems with data matching. In clinical database this issue is common in free-text fields such as investigators’ assessments. The special characters might be entered by mistake or in order to make the entry more clear at the input stage. However once the text is structured within a dataset this adjustment does not provide any benefit. It is though relatively simple to build-in a check for hidden characters in the data validation process. Example of such a check is presented in the application part of this paper.

Spelling mistakes are more difficult to deal with because they require large matrices or well-established algorithms enabling to consider numerous patterns for erroneous entries for a given word. A standardization technique would replace various spelling by a single unified word (Winkler, 2004, p. 538). Non-standardized entries can occur for example on the results of laboratory tests entered as text field. We can imagine three different texts strings e.g. ‘Neg’ ‘neg’ and ‘negative’ representing the same meaning. In order to standardize various entries of the same word we can use reference dictionary containing possible versions of a given spelling and replace the actual by the standard expression. Some more complicated techniques, like clustering, can be considered for the purpose of spelling checks, see for example (Elder, Miner, & Nisbet, 2012, p. 48).

The checks for numeric data are focused on the question: how plausible a given result is? Statistical theory offers a set of analytical (3σ and interquartile range rule) and graphical tools (e.g. box and whisker plot) for detection of outliers. The 3σ rule is based on the feature of the normal distribution which is that 99.7% of the values is within the range of the three standard deviations. For a normally distributed variable, any value $|x| > 3\sigma$ should be checked as it is a potential outlier. Similarly we can refer to quantile rule in statistics which states that outlier is a value that lies below Q1-1.5IQR or above Q3+1.5IQR, where interquartile range IQR = Q3 - Q1, and Q1, Q3 are lower and upper quartiles respectively.

A check using the normal approximation can be applied to vital signs measurements. Figure 5 shows the histogram of systolic blood pressure measurements flagging out the ones which are not in line with the expectation under normality assumption and thus likely to be erroneous. The program code used to create the graph is based on three main blocks. The first block is to select the variables of interest from the final transposed dataset (see Figure 4). The values need then to be standardized to identify outliers. The last block is to calculate the counts and draw the histogram. The source code used to produce Figure 5 is as follows:

```
*Select measurements of interest (systolic blood pressure);
data a01;
 set o;
 std=ParamVl;
 where index(paramcd,'SYS')>0;
run;

*Standardize the values;
proc standard data=a01 out=a02 mean=0 std=1;
 var std;
run;

*Calculate the frequencies;
proc freq data=a02 noprint;
 tables ParamVl*std / out=a03;
run;

*Prepare data for dot plot;
data a04;
 set a03;
 *Flag the outliers according to 3-sigma rule;
 if abs(std)>3 then do;
  OutVal=ParamVl;
 end;
 else Val=ParamVl;
 *Multiply records to create dot plot;
 if count=1 then do;
  i=count;
  output;
 end;
 else if count>1 then do;
  do i=1 to count;
   output;
  end;
 end;
 label val="Values" i="Frequency";
run;

*Draw the histogram using dot plot;
proc sgplot data=a04;
 scatter x=Val     y=i / markerattrs=(symbol=CircleFilled size=5 color=blue);
 scatter x=OutVal  y=i / markerattrs=(symbol=DiamondFilled size=6 color=red);
run;
```

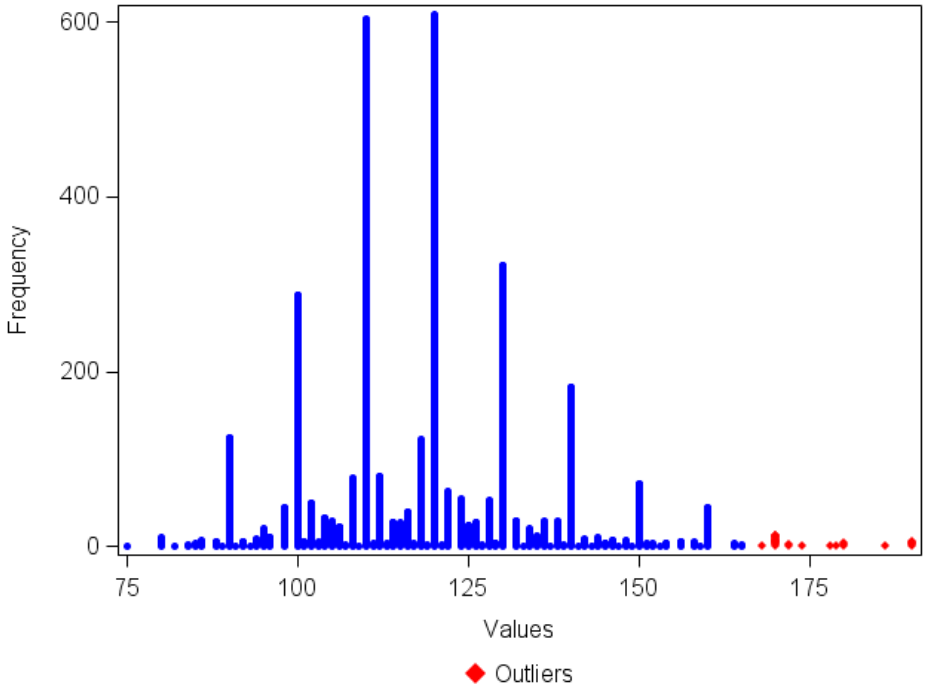


Figure 5 Histogram of systolic blood pressure measurements in PHYSEXA and PHYSEXE datasets from the study NIDA-CPU-0016 (2014)
Source: Own elaboration.

The checks described above are applicable to the result itself and are helpful in detection of erroneous data. Numeric data can be investigated far more with use of other techniques one of which is distribution of first digit referred as Benford's Law, see (Judge & Schechter, 2009). According to the Benford's Law, first digit (1,2,…,9) follow a specified distribution as long as the number of values is large. The probability density function of the theoretical distribution is expressed by the formula (Judge & Schechter, 2009, p. 2):

$$P(D = d) = \log_{10}\left(1 + \frac{1}{d}\right), \; d = 1,2,\ldots,9. \qquad (1)$$

The check for distribution of first digit is based on Benford's observation that it would be unusual for a distribution of values from a large dataset to not follow the theoretical distribution (1). The difference between the theoretical counts of first digit and the empirical counts can be inspected graphically (see Figure 6). In survey data applications, we can refer to a statistical goodness-of-fit test to assess whether the observed and theoretical counts are generated out of different populations, see (Judge & Schechter, 2009, p. 5). The null hypothesis in the goodness-of-fit test assumes that the cumulative distribution functions of the empirical distribution $F(x)$ and theoretical distribution $F_0(x)$ are equal:

H_0: $F(x)=F_0(x)$, versus the alternative that:

H_1: $F(x)\neq F_0(x)$.

The test statistic compares the observed counts (n_i) with the counts under the null hypothesis ($n \cdot p_i$) in each of the m categories (i = 1,2,…,m), see (Mittelhammer, 2013, pp. 668-71):

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \qquad (2)$$

where n is the sample size, n_i is the empirical count in the ith category and p_i represents the probabilities of the theoretical distribution. This is multinomial probability distribution with parameters ($p_1,p_2,\ldots,p_m$) and $\sum_{i=1}^{m} p_i = 1$. Statistic (2) follows χ^2 distribution with m-1 degrees of freedom, where m represents the number of categories of the variable. The rejection region is defined as $P(\chi^2 \geq \chi^2_{\alpha,m-1}) = \alpha$, while the *p-value* is $P(\chi^2 \geq \chi^2_e) = P$ where χ^2_e represents the value of the test statistic (2). In case of the first digit distribution m = 9. The check for the distribution of first digit has been exemplified in the application section.

Dates and times are special types of numeric data. In practice dates and times are often partial. It is then important to consider a technique of filling-in the missing parts of the dates and times at the data cleaning stage. The issue with partial dates is commonly observed in adverse event, medical history and concomitant medication data where it is not always easy to establish the exact dates when a particular event has started or ended. Moreover this special type of data allows checking

for unusual and potentially fraudulent events in the process under analysis. In practice it could be for example a specific procedure or measurement carried out outside of the protocol scheduled time point. The unusual dates and times can be used to detect misconduct in a procedure under analysis like for example intake of a prohibited medication in the investigational product treatment period. This kind of checks are vital for identification of protocol deviations based on which the analysis sets are specified and the actual analysis is performed. In other applications we may consider a financial operation performed in very early morning. This could be perceived as unusual time for this kind of operation and indicator of a fraud risk on a consumer account[2]. In this application the dates and times can be inspected graphically in order to identify any outliers, e.g. in clinical trial a well-planned measurement carried out during the night. More broadly in quality control systems it could be a performance registered wrongly at the day when the company was not operating.

The two last data issues do not refer to any special data type. Statistical research practice shows that missing data is inherent in the data collection process. The missing data problem can be viewed as a data quality problem and an estimation problem. Missing data could occur because of the design of a study, for example when the collection of the specific data is not straightforward and it requires an important input form the investigator. An example would be the collection of laboratory data in clinical trials when reporting includes results and some other characteristics like ranges. In such a case we can say that missing data is a quality problem, which can be reduced by the checks at the source of data.

On the other hand in empirical studies we deal with data missing due to events related to the responders behavior and not to the data collection process itself. Still this is a data quality problem but a data check would not provide any amendment to the database. Our responders might refuse to answer some questions or they could miss a survey for some reason. This results in nonresponse bias whenever the populations of responders and non-responders differ. In biostatistics, missing values might be related to the negative outcome of the parameter of interest. For example a worsening heath status of a patient could discourage them to go to the clinic for a scheduled visit. The missing assessments would not be completely random in this case but in fact the missingness would be related directly to the value of the assessment itself. Special considerations would need to be taken into account to minimize the encountered bias. Similarly we may think of an example from other research areas. In marketing research, it's likely that unsatisfied customers would be more inclined to fill-in a service questionnaire than the satisfied ones (Hoerl & Snee, 2012, pp. 151-2). In such case missing values are considered an estimation problem and thus we usually refer to some imputation techniques in order to reduce the nonresponse bias. The topic of missing data and imputation methods is extensively revised and exemplified in (Little & Rubin, 2002). An overview of the approaches to the missing data can be found in (Korczyński, 2014).

An important feature of a database is to have the same information provided only once, see (Winkler, 2004, p. 533). The duplicate records can cause issues at analysis stage when we need to combine datasets as well as in business communication. In clinical trials the efficacy analysis might require to combine the results of several different assessments across visits. Duplicate assessments would lead to merging issues. In industries like Banking and Telecommunications specific automatic reports are often provided to customers. Duplicates in the address databases are generating unnecessary cost of sending the same message more than one time to one customer. This also can affect the image of the company. In order to improve the data quality a company needs to introduce an explicit rule of storing data in one place only and follow a validation procedure when data is created (Laursen & Thorlund, 2010, p. 180).

EXAMPLE OF AN APPLICATION FOR CLINICAL TRIAL DATA

The outlined technique of checking data has been applied to the data from clinical trial conducted by the National Institute on Drug Abuse (NIDA). Specifically, data from NIDA-CPU-Methylphenidate-0001 were included. The input database contained 33 datasets with 215,053 observations and 2,306 variables in total, see Appendix B. The input datasets have been transposed and set with the use of the *TransposeAllLib* macro program as shown in the Appendix A.

The transposition of the datasets took 14 minutes and 15 seconds. The output dataset composed of 2,555,301 records, has been used to apply two example checks on the data: check for hidden characters and check for first digit distribution for the numeric values from the laboratory dataset LABSA. The checks have been carried out using the respective macro programs: *FindHidChars* and *FirstDigitDist* as shown in the Appendix A.

There were no hidden characters found in any of the datasets. The results of the check for the first digit distribution are illustrated on Figure 6. Shadowed bars represent theoretical distribution defined by Benford's Law. White bars represent the empirical distribution of first digit. The white and shadowed tops indicate the discrepancies between the two distributions. The figure shows also the value of the χ^2 statistic in the test for goodness-of-fit with the associated *p-value*. The results do not indicate statistically significant deviation of the empirical counts, of the numeric entries of the LABSA dataset from the theoretical counts under Benford's Law. As we can see on the graph the true and expected counts are not much different.

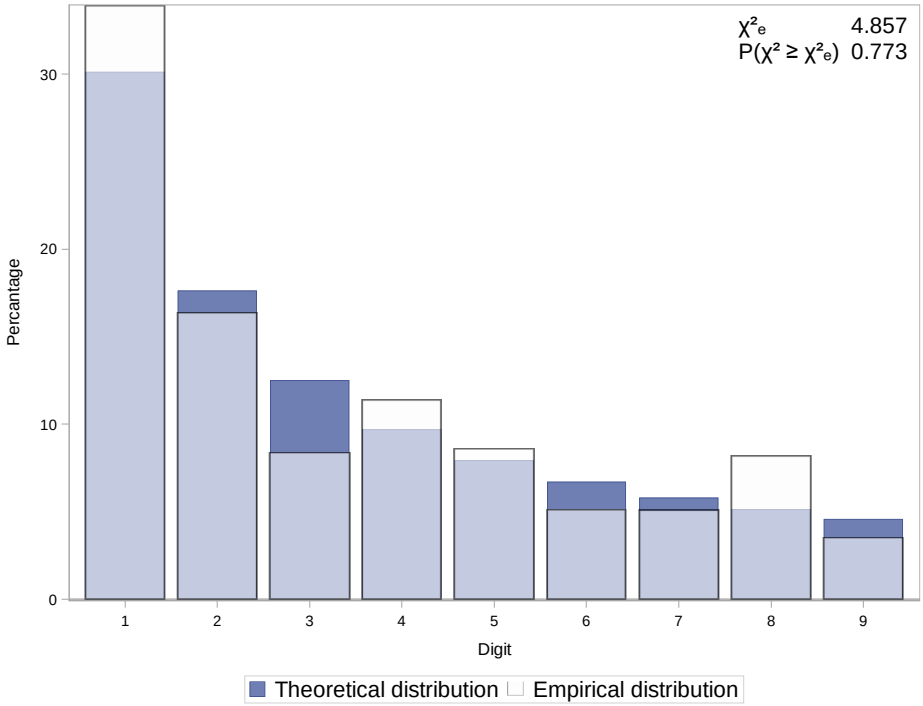


Figure 6 Theoretical and empirical distribution of first digit of numeric values in LABSA dataset from the study NIDA-CPU-0016 (2014)
Source: Own elaboration with use of the *FirstDigitDist* program from the Appendix A.

CONCLUSION

Successful analytics requires good quality data, see (Isson & Harriott, 2013, p. 62; McQuinggan & Sapp III, 2014, pp. 62-4). The data quality affects the analytical solutions. In data research we should however expect data issues to occur. It is important to consider a validated procedure enabling us to maintain the data quality, see (Laursen & Thorlund, 2010, p. 180). In business applications, whenever the data collection process is complicated we can expect that Data Management will take the responsibility of data cleaning. The aim of this paper was to highlight how the analytical team i.e., the final user of the data can support DM. The input from the analyst is crucial as it outlines the overall reporting requirements.

This idea was exemplified by a technique, which can be used for the data checks. The technique is based on data combining and scanning for issues, according to the pre-defined rules, specified with the input form final data users. Detected issues are then corrected before the data is transferred to the analyst. This procedure has been applied to an example database from a clinical trial. The example shows that it could be considered as an efficient way of checking for data quality in the database of moderate size. The implementation of such a technique would reduce the amount of resource needed for both data manager and analyst. Data manager would need to spend less time on the queries and the analyst will have access to the cleaner data, which will boost up the analysis process.

The outlined procedure requires further insight into the types of data checks. Furthermore, as it is designed to small and moderate size databases some more efficient methods can be considered for large database applications.

REFERENCES

[1] For larger databases the step setting the transposed datasets into one final dataset can be omitted and any checks could be carried out for each of the transposed dataset separately by looping over these datasets at each data check.

[2] The idea of fraud detection with application to Telecommunications industry can be found in (Reis Pinheiro & McNeill, 2014, pp. 120-8, 178-88).

Elder, J., Miner, J., & Nisbet, B. (2012). Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications. Waltham: Elsevier Inc.

Hoerl, R., & Snee, R. (2012). Statistical Thinking. Improving Business Performance. Hoboken: John Wiley & Sons.

Isson, J. P., & Harriott, J. S. (2013). Win with Advanced Business Analytics. Hoboken: John Wiley & Sons.

Judge, G., & Schechter, L. (2009). Detecting Problems in Survey Data Using Benford's Law. The Journal of Human Resources, 44 (1), 1-24.

Korczyński, A. (2014). Review of methods for data sets with missing values and practical applications. Silesian Statistical Review, 83-104.

Laursen, G. H., & Thorlund, J. (2010). Business Analytics for Managers. Taking Business Intelligence Beyond Reporting. Hoboken: John Wiley & Sons.

Little, J. A., & Rubin, D. (2002). Statistical Analysis with Missing Data. Hoboken: John Wiley & Sons.

McQuinggan, J., & Sapp III, A. W. (2014). Implement, Improve and Expand Your Statewide Longitudinal Data System. Creating a Culture of Data in Education. Hoboken: John Wiley & Sons.

Mittelhammer, R. C. (2013). Mathematical Statistics for Economics and Business. New York: Springer.

NIDA. (2014, 12 04). National Institute on Drug Abuse. Data Share Website. Retrieved 08 21, 2016 from NIDA-CPU-0016: Double-Blind, Placebo-Controlled Assessment Of Potential Interactions Between Intravenous Methamphetamine And Osmotic-Release Methylphenidate (Oros-Mph): https://datashare.nida.nih.gov/show-data-for-study/nida-csp-999.

Ohlhorst, F. J. (2013). Big Data Analytics Turining Big Data Into Big Money. Hoboken: John Wiley & Sons.

Reis Pinheiro, C. A., & McNeill, F. (2014). Heuristics in Analytics. A practical Perspective of What Influences Our Analytical Word. Hoboken: John Wiley & Sons.

Winkler, W. E. (2004). Methods for evaluating and creating data quality. Information Systems, 29, 531-50.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Adam Korczyński
QuintilesIMS
ul. Gottlieba Daimlera 1
02-460 Warszawa
Work Phone: +48 22 430 31 57
Email: Adam.Korczynski@QuintilesIMS.com

APPENDIX A – DATA CLEANING MACRO PROGRAM, WRITTEN IN SAS V9.2

```
*Macro programs for standard data checks:
1. LibContents - to check the contents of a library,
2. DataLong - to transpose a single dataset,
3. TransposeAllLib - to transpose all datasets in a library,
4a. FindHidChars - to search for hidden characters (byte1-31),
4b. FirstDigitDist - to check the distribution of first digit.;

*#1. Macro to check the contents of a library;
%macro LibContents(
Lib= /*Library name (e.g. work) and the default name of the output dataset with
description of the library contents*/
,DSets= /*List of the datasets to filter for (e.g. dm ex)*/
,OutSet= /*Name of the output dataset*/
,DelTemp=/*Set to Y if temporary datasets are to be deleted*/
);

*# Check if the name of the output dataset is specified;
%if %length(&OutSet.)=0 %then %let OutSet=&Lib.;
*# Create list of datasets to filter for;
%if %length(&DSets.)>0 %then %do;
 data _null_;
  NOfDSetsL=countw("&DSets.");
  DSetsList='"'||tranwrd(upcase("&DSets."),' ','" "')||'"';
  call symput('DSetsList',DSetsList);
  call symput('NOfDSetsL',NOfDSetsL);
 run;
%end;
proc sql noprint;
 create table _datasets as select
  *
 from dictionary.tables
 where libname=upcase("&LIB")
 %if %length(&DSets.)>0 %then and memname in (&DSetsList.);;
*# Check the number of datasets from the list found in the library;
 select count(*) into :NOfDSetsO from _datasets;
 create table _variables as select
  a.*
  from dictionary.columns a
  where libname=upcase("&LIB")
  and memname in (select distinct memname from _datasets)
  ;
 create table &OutSet. as select
  a.libname, a.MemName, a.MemLabel, a.crdate as MemCrDate
 ,a.nobs as MemNobs, a.nvar as MemNVar, a.encoding as MemEncoding
 ,b.varnum as VarNum, b.name as VarName, b.label as VarLabel
 ,b.type as VarType, b.length as VarLength, b.format as VarFormat
     from _variables b left join _datasets a
     on (a.libname = b.libname and a.memname = b.memname)
        order by a.MemName, b.varnum
     ;
quit;

*# Inform if the dataset from the list is not in the library;
%if %length(&DSets.)>0 %then %do;
 %if &NOfDSetsO. = 0 %then
 %put ERR%str(OR):: Datasets from the list not found in the   library.;
 %else %if &NOfDSetsO. > 0 and &NOfDSetsL. > &NOfDSetsO. %then
 %put WARN%str(ING):: Only some datasets from the list were found in the library.;
%end;
*#Delete temp datasets from work library;
%if &DelTemp. = Y %then %do;
ods exclude all;
 proc datasets library=work;
 delete _datasets _variables;
 run;
ods select all;
%end;
%mend LibContents;
```

```
*#2. Macro to transpose data to have one value for each parameter;
%macro DataLong(InpSet=  /*Input dataset*/
                         ,byVars= /*By variables*/
                         ,Params= /*List of the parameters*/
                         ,OutSet= /*Output dataset*/
                         ,OutSrt= /*Sorting variables for output dataset*/
                         ,DelTmp=/*set to Y to delete temporary datasets*/
);
data _tmpdl01;
 set &InpSet.;
run;

%let k=1;
%let ParamList=;
%do %until(%nrbquote(%scan(&Params.,&k,' '))=);
%let Param&k=%scan(&Params.,&k,' ');
 proc sort data= _tmpdl01
           out = _tmpdl02 (keep = &byVars. &&Param&k. where = (^missing(&&Param&k.)))
  nodupkeys;
  by &byVars.;
 run;

*Check if there are any non-missing values - if no skip processing;
proc sql noprint;
 select count(*) into :NonMiss from _tmpdl02;
quit;
%if &NonMiss. = 0 %then %goto NextVar;
*#Get the type of the parameter;
 data _null_;
  set _tmpdl02;
  call symput('ParamTp',vtype(&&Param&k.));
 run;

data _tmpdlParam&k (where = (ParamVlC^='.') drop = &&Param&k.);
       format &byVars.;
       length ParamCd $40 ParamLb $100 ParamFt $20 ParamTp $1 ParamVlC ParamVlCD $1000;
   set _tmpdl02;
   ParamCd=vname(&&Param&k.);
   ParamLb=vlabel(&&Param&k.);
   ParamFt=vformat(&&Param&k.);
   ParamTp=vtype(&&Param&k.);
   %if &ParamTp. = C  %then %do;
    ParamVl = 1;
    ParamVlC =&&Param&k.;
    ParamVlCD=trim(left(putc(&&Param&k.,ParamFt)));
      %end;
      %else %if &ParamTp. = N %then %do;
       ParamVlCD=trim(left(putn(&&Param&k.,ParamFt)));
       if anyalpha(ParamVlCD)>0 then do;
        ParamVl = 1;
        ParamVlO = &&Param&k.;
       end;
       else if anyalpha(ParamVlCD)=0 then do;
        ParamVl = &&Param&k.;
        ParamVlO = ParamVl;
       end;
       ParamVlC = trim(left(put(ParamVl,best.)));
      %end;
run;

%if %length(&ParamList.)=0 %then %do;
 %let ParamList = _tmpdlParam&k.;
%end;
 %else %if %length(&ParamList.)>0 %then %do;
 %let ParamList = &ParamList. _tmpdlParam&k.;
%end;

%NextVar: %let k = %eval(&k+1);
%end;
```

```
%let k = %eval(&k - 1);

data _tmpdl03;
 set &ParamList.;
run;

%if %length(&OutSrt.) > 0  %then %do;
 proc sort data=_tmpdl03 out=&OutSet.;
  by &OutSrt.;
 run;
%end;
%else %do;
 data &OutSet.;
  set _tmpdl03;
 run;
%end;

*#Delete temp datasets from work library;
%if &DelTmp. = Y %then %do;
ods exclude all;
proc datasets library=work;
 delete _tmpdl:;
run;
ods select all;
%end;
%mend DataLong;

*#3. Macro to transpose datasets from a library;
%macro TransposeAllLib(InpSet= /*Input dataset*/
                       ,byVars= /*By variables. The first variable is key ie it
must appear in the dataset in order to consider it for transposition e.g. SUBJID. Must
contain sequence variable other than key ID marked by # e.g LINE#*/
                       ,OutSet= /*Output dataset*/
                       ,DelTmp= /*set to Y to delete temporary datasets*/
);

%let StartTime = %sysfunc(datetime());
%LibContents(Lib=&InpSet.,DelTemp=Y);

*#Create a string with ID variables list;
%if %length(&byVars.)>0 %then %do;
%put &byVars.;
*#Identify the sequence variable [marked by "#" at the end];
%let pos1 = %sysfunc(find("&byVars.",#,-1000));
%let SeqAsLast = %qsubstr(&byVars.,1,&pos1.-2);
%let pos2 = %sysfunc(anyspace("&SeqAsLast.",-1000));
%let SeqVar = %qsubstr(&SeqAsLast.,&pos2.);
%put &SeqVar.;
*#Update ID variables to drop Seq indicator;
%let byVars=%sysfunc(compress(&byVars.,#));
%put &byVars.;
 data _null_;
  List='"'||tranwrd(compbl(upcase("&byVars.")),' ','" "')||'"';
   call symput('byVarsQ',List);
  run;
 %put &byVarsQ.;
%end;
*#Extract each single ID variables and save as macro variables;
%if %length(&byVars.)>0 %then %do;
 %let k=1;
 %do %until(%nrbquote(%scan(&byVars.,&k,' '))=);
       %let Value&k=%upcase(%scan(&byVars.,&k,' '));
             %put Value&k = &&Value&k;
             %put &k;
             %let k = %eval(&k+1);
 %end;
 %let NbyVars=%eval(&k-1);
 %put &NbyVars.;
%end;
```

```
*#Check if ID variables are present in the datasets;
proc sql;
 create table _tmpdls01 as select
       *
 %do i=1 %to &NbyVars.;
       ,sum(varname in ("&&Value&i")) label="&&Value&i." as KeyVar&i.
 %end;
 from &InpSet.
 group by memname;
quit;

*#Transpose only datasets with key ID variable present and with at least one record
[skip ID variables from the list];
proc transpose
 data = _tmpdls01 (where = (varname not in (&byVarsQ.) and memnobs>0 and KeyVar1=1))
 out =  _tmpdls02;
 by libname memname KeyVar:;
 var varname;
run;

*#Create the specification of the datasets in the library;
data VarSpec;
 length DVList List $5000;
 set _tmpdls02 end=last;
 Loc = trim(left(libname))||'.'||trim(left(memname));
 List = catx(' ', of col:);
 DVList = trim(left(memname))||'['||List||']';
 DSetIdN+1;
 DSetId='DS'||trim(left(put(DSetIdN,best.)));
 DSetLoc='LOC'||trim(left(put(DSetIdN,best.)));
 NKeyVars=sum(of KeyVar:);
 keep Loc DVList List DSetIdN DSetId DSetLoc KeyVar: NKeyVars memname libname;

       *Output variables and locations;
 call symputx(DSetId,List);
 call symputx(DSetLoc,Loc);
 if last then call symputx('NSets',DSetIdN);
run;

%let RunIndex=1;
%do i=1 %to &NSets.;
%put ########### Processing Dataset: &&loc&i. ###########;
 data _tmpdlIn01;
  set &&loc&i.;
  UniqueKey=1;
 run;
 *Bring in domain name and library;
 data _tmpdlname;
 set VarSpec (where = (DSetIdN=&i.) keep = DSetIdN memname libname);
 UniqueKey=1;
 run;
 data _tmpdlIn02;
  merge _tmpdlIn01 (in=a) _tmpdlname (in=b);
  by UniqueKey;
 run;
 %do j=2 %to &NbyVars.;
  proc sql noprint;
       select KeyVar&j. into :IndValKeyVar&j. from VarSpec where DSetIdN=&i.;
  quit;

%if &&IndValKeyVar&j.=0 and %upcase(&&Value&j.)^= %upcase(&SeqVar.) %then %do;
*#If second ID variable is missing;
 data _tmpdlIn02;
  set _tmpdlIn02;
  &&Value&j="";
 run;
%end;
```

```
%else %if &&IndValKeyVar&j.=0 and %upcase(&&Value&j.)= %upcase(&SeqVar.) %then %do;
*#If line ID variable is missing then set to dummy sequence;
 proc sort data=_tmpdlIn02 out=_tmpdlIn03;
  by &Value1.;
 run;
 data _tmpdlIn02;
  set _tmpdlIn03;
  by &Value1.;
  retain &SeqVar.;
  if first.&Value1. then &SeqVar.=1;
  else &SeqVar.+1;
 run;
%end;

%end;

%DataLong(InpSet=_tmpdlIn02
               ,byVars=&byVars. memname libname
               ,Params=&&ds&i.
               ,OutSet=_tmpdlOutInd
               ,OutSrt=&byVars.
               ,DelTmp=);
%if &RunIndex.=1 %then %do;
 data &OutSet.;
  set _tmpdlOutInd;
 run;
%end;
%if &RunIndex.>1 %then %do;
 data &OutSet.;
  set &OutSet. _tmpdlOutInd;
 run;
%end;
%let RunIndex = %eval(&RunIndex+1);
%end;

*#Delete temp datasets from work library;
%if &DelTmp. = Y %then %do;
 ods exclude all;
  proc datasets library=work;
   delete _tmpdl: ;
  run;
 ods select all;
%end;

data _null_;
 duration = datetime()-&StartTime.;
 put 'Total duration:' duration time13.2;
run;
%mend TransposeAllLib;

*# 4a. Macro to find hidden characters in a variable;
%macro FindHidChars(InpSet= /*Input dataset*/
                                ,Var=/*Variable to check*/
                                ,Whr=/*Optional filter*/
                                ,OutSet= /*Output dataset*/);
data &OutSet. (where=(sum > 0));
 set &InpSet.;
 array _tmpsc {*} _tmpsc1-_tmpsc31;
  do i=1 to dim(_tmpsc);
       _tmpsc{i} = index(&Var.,byte(i));
  end;
  sum=sum(of _tmpsc1-_tmpsc31);
  %if %length(&whr.)>0 %then where &Whr.;;
run;
%mend FindHidChars;
```

```
*#4b. Macro to check the distribution of first digit, Benfords Law;
%macro FirstDigitDist(InpSet= /*Input dataset*/
             ,Var=/*Variable to check*/
                                ,Whr=/*Optional filter*/
                                ,OutSet= /*Output dataset*/
                                ,Plot=/*Set to Y for plot of theroretical
                                              versus emprical distribution*/
                                );
*#Derive d (digit) and filter data;
data _tmp01 (where=(d>0));
 set &InpSet. %if %length(&whr.)>0 %then (where=(&Whr.));;
 d=input(substr(compress(&Var.,'-'),1,1),best.);
run;
*#Count digits;
proc freq data=_tmp01 noprint;
 tables d / out=_tmp02(rename=(percent=e));
run;

*#Compare against Benfords first digit distribution;
data &OutSet.;
 set _tmp02 end=last;
 t=log10(1+1/d)*100;
 Dif=(e-t)**2/t;
 sum+dif;
 if last then do;
  p=1-probchi(sum,8);
  call symputx('Chi2e',putn(sum,'5.3'));
  call symputx('p',putn(p,'5.3'));
 end;
run;

%if %upcase(&Plot.)=Y %then %do;
 *#Plot empricial and theoretical distributions;
 %let _Chi2=(*ESC*){unicode '03C7'x '00B2'x};
 %let _ge = (*ESC*){unicode '2265'x};
 %let _sube = (*ESC*){unicode '2091'x};
 proc sgplot data=&OutSet.;
  vbar d / response=t name='t'
legendlabel='Theoretical distribution';
  vbar d / response=e transparency=.5 name='e' legendlabel='Empirical distribution';
  yaxis label='Percantage';
  xaxis label='Digit';
  keylegend 't' 'e' / down=1;
  inset
   ("&_Chi2.&_sube."="&chi2e." "P(&_Chi2. &_ge. &_Chi2.&_sube.)"="&p.") / position=ne;
 run;
%end;

%mend FirstDigitDist;

*#Example of the use of the macro for checks on the NIDA datasets;
*#Create transposed dataset;
%TransposeAllLib(InpSet=raw
 ,byVars=patient study line#
 ,OutSet=o
 ,DelTmp=);

*#Simple scan check;
%FindHidChars(InpSet=o
 ,Var=paramvlc
 ,Whr=
 ,OutSet=o1);

*#Visual and statistical check;
%FirstDigitDist(InpSet=o
 ,Var=paramvlc
 ,Whr= paramtp eq 'N' and memname eq 'LABSA'
 ,OutSet=o2
 ,Plot=Y);
```

APPENDIX B – DATASETS FROM NIDA STUDY

Table 1 List of datasets from the study NIDA-CPU-Methylphenidate-0001

Number	*Dataset Name*	*Number of Observations*	*Number of Variables*
1	*AEA*	9987	121
2	*AEE*	7500	201
3	*BKG*	736	21
4	*CONMEDA*	9964	247
5	*CONMEDE*	1915	247
6	*CONSENT*	736	15
7	*COORDREP*	10139	104
8	*CRAV*	737	9
9	*DHX*	736	51
10	*DOSEA*	69658	13
11	*DOSEE*	52231	15
12	*DSM3*	0	15
13	*ECGA*	1820	37
14	*ECGE*	471	37
15	*ENRL*	824	33
16	*FORM20*	184	10
17	*FUP*	736	17
18	*GRP*	3066	9
19	*GRS*	3064	9
20	*LABSA*	3791	57
21	*LABSE*	744	50
22	*MHX*	736	425
23	*OTCMEDS*	1915	247
24	*PHYSEXA*	3076	48
25	*PHYSEXE*	691	39
26	*PSYCH*	9977	24
27	*SAEA*	77	18
28	*SAEE*	44	19
29	*SURA*	9991	75
30	*SURE*	7502	71
31	*TERMA*	736	9
32	*TERME*	309	9
33	*TXGRP*	960	4
Total		215053	2306

Source: Own elaboration based on datasets from (NIDA, 2014).