EXTENDING THE USE OF ADMINISTRATIVE DATA IN ROMANIAN OFFICIAL STATISTICS

VAT DATA

Outline

I. Introduction

The use of the administrative data for the statistical production process has become a priority in the statistical system in Romania, deriving from the need to exploit information already existing in the public central administration databases and aimed at reducing the response burden of the units included in the classic statistical surveys. The Value Added Tax (VAT) dataset is one of the most important administrative data used in the production of official statistics. This report describes methods for detecting and correcting errors, validating and editing the turnover variable as it is found in the VAT administrative dataset. The statistical methods and techniques used are derived from the research literature and best practices in this field.

II. Quality indicators

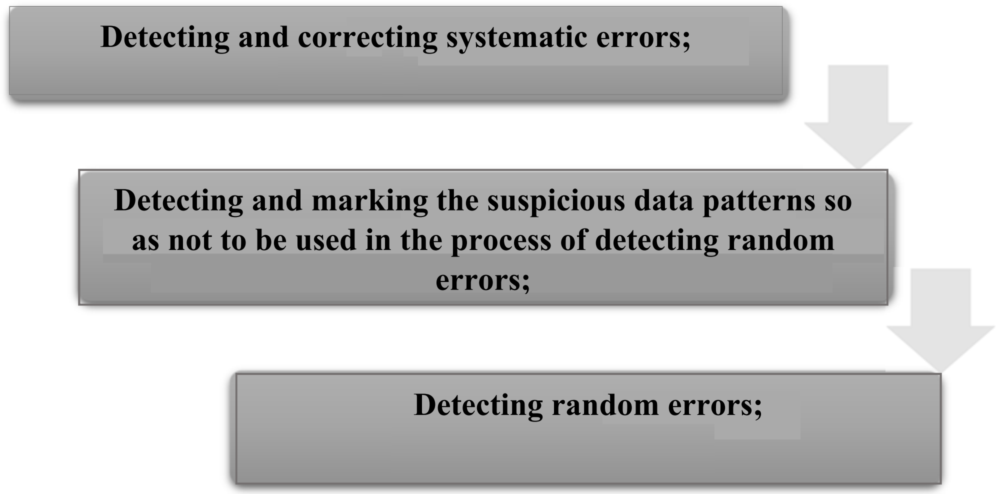
The list of quality indicators is useful when planning to start using administrative data as a replacement for, or to supplement, survey data. In this scenario, the indicators can be used to assess the feasibility of moving to administrative data, and the impact on output quality. The indicators have been developed so that a low indicator score denotes high quality, and a high indicator score denotes low quality.

Indicator	**Description**
Item non-response (% of units with missing values for key variables)	This indicator should be calculated for each of the key variables. $$\frac{Number\ of\ units\ in\ administrative\ data\ with\ missing\ values\ for\ the\ variable\ X}{Number\ of\ units\ in\ administrative\ data}\ x\ 100\%$$
Misclassification rate	This indicator provides information on the proportion of units in the administrative data which are incorrectly coded. For simplicity and clarity, activity coding as recorded on the Business Register (BR) is considered to be correct. If the activity code from the administrative data is not used by the NSI (e.g. if coding from BR is used), this indicator is not relevant. $$\frac{Number\ of\ units\ in\ administrative\ data\ with\ NACE\ different\ from\ BR}{Number\ of\ units\ in\ administrative\ data}\ x\ 100\%$$
Undercoverage	This indicator provides information on the undercoverage of the administrative data. That is, units in the reference population that should be included in the administrative data but are not (for whatever reason).

	This indicator could be calculated based on the number of relevant units in the business register (BR). $$\frac{Number\ of\ units\ in\ BR\ but\ NOT\ in\ administrative\ data}{Number\ of\ units\ in\ BR}\ x\ 100\%$$
Overcoverage	This indicator provides information on the overcoverage of the administrative data. That is, units that are included in the administrative data but should not be. This indicator should be calculated based on the number of relevant units in the business register (BR). $$\frac{Number\ of\ units\ in\ administrative\ data\ but\ NOT\ in\ BR}{Number\ of\ units\ in\ BR}\ x\ 100\%$$
Size of revisions from the different versions of the admin data RAR – Relative Absolute Revision	With this indicator it is possible to understand the impact of the different versions of administrative data on the results for a certain reference period. When data is revised based on other information (e.g. survey data) this should not be included in this indicator. If only one version of the administrative data is received, this indicator is not relevant. $$\frac{\sum_{t=1}^{T}\lvert X_{Lt}-X_{Pt}\rvert}{\sum_{t=1}^{T}\lvert X_{Pt}\rvert}\ x\ 100;\ X_{Pt}=first\ data;\ X_{Lt}=latest\ data$$

III. Methods for Detecting Errors in VAT Data

Steps to follow in order to detect errors in VAT Data


Detecting and correcting systematic errors;
Detecting and marking the suspicious data patterns so as not to be used in the process of detecting random errors;
Detecting random errors;

Three categories of errors have been identified that are likely to be found in administrative data:

a) Systematic errors
b) Suspicious data patterns
c) Random errors

1. Methods for detecting systematic errors

Some businesses might report by mistake their figures in RON instead of thousands of RON, and vice versa. Assuming that the business has not made the same mistake over two consecutive periods, such errors can be often detected by comparing with previous VAT returns for the same business, using the next formula:

$$A < \frac{current\ VAT\ data}{previous\ VAT\ data} < B$$

The values of the parameters A and B determine if a figure is likely to be a unit error or not. These values are usually fine-tuned using past data.

2. Methods for detecting suspicious data patterns

When longitudinal data are available for analysis, it is often possible to identify errors in VAT data by considering the pattern of reported VAT data over a period. These patterns can be summarised as follows:

a) Zero VAT data values in three quarters, positive VAT data value in the other quarter.
b) Zero VAT data value in one quarter, positive VAT data values in the other three quarters.
c) Same VAT data values in all four quarters.
d) Same VAT data values for three quarters, a different (positive) VAT data value in the other quarter.
e) Negative VAT data value in any of the quarters.

3. Methods for detecting random errors

These methods adopt either of the two main approaches below:

a) Compare the returned value in the current period to the returned value in the previous period for the same business.
b) Compare the returned value for a business with the returned value of similar businesses in the same period. For example, one can calculate the mean or median of returns for similar businesses. This would give a single number that is representative of a group of businesses that we can use for comparison. Some criteria in order to group businesses together into classes are the NACE industry and the employment size bands.

A. Quartile Distances

This method detects unusually large or small VAT data values by locating extreme values in the distribution of VAT data within a particular industry and size class. The method identifies suspicious businesses as those with VAT data values greater than a specified multiple of inter-quartile distances from the third quartile, Q3 (for large VAT data values) or less than the same multiple of inter-quartile distances from the first quartile, Q1 (for small VAT data values). More formally, suspicious VAT data values are identified as follows:

if
VAT return > Q3 + [C × (Q3 – Median)]
or
VAT return < Q1 – [C × (Median – Q1)]
then treat as suspicious

The quartiles and median are derived from the VAT data within the industry and size class for the period under consideration. The parameter C is derived from analysing past data to find a threshold that successfully identifies extreme values.

B. Period on period ratios

This method involves calculating period on period ratios for each business based on the contribution of that business's VAT data value to its class. The class would commonly be defined as a cross-classification of NACE, size and periodicity. The period on period ratios are called test ratios. Any business with a test ratio above a predefined threshold is judged as suspicious. The method is described below:

$$Score = \frac{VAT\ return}{Median\ VAT\ return\ in\ class}$$

$$TestRatio = \begin{cases} \frac{Score_t}{Score_{t-1}} & if\ Score_t > Score_{t-1} \\ \frac{Score_{t-1}}{Score_t} & otherwise \end{cases}$$

$Score_t$ is the value of Score in period t and $Score_{t-1}$ is the value of Score in period t $-$ 1.

C. Comparison with reporting history for the business

This method compares the current VAT data figure with previous historic values for the same business to determine likely suspicious values.

If
VAT return > 10000 RON
and
VAT return > 10 × mean VAT return for the business in the past 12 months
Then treat as suspicious

D. Hidiroglou-Berthelot

This method is based on the simple ratio between the VAT return for a business in the current period and the VAT return for the same business in the previous period. The method adds some refinements to overcome various drawbacks of using a simple ratio and also to allow for the size of the business to be taken into account. The method is defined as follows:

$$r = \frac{current\ VAT\ return}{previous\ VAT\ return}$$

$$If\ r < median$$

$$t = \frac{r - median}{r}$$

$$otherwise$$

$$t = \frac{r - median}{median}$$

E = t * max (current VAT Turnover, previous VAT Turnover) * V

The parameter V can take any value between 0 and 1. A value of 0 results in every business having the same importance, whereas a value of 1 gives greatest importance to businesses with higher VAT data values. Calculate the first and third quartiles (Q1 and Q3) and the median (Q2) of the E values. Then calculate:

$$d_{Q1} = max[(Q2 - Q1), |A\ x\ Q2|]$$

$$d_{Q3} = max[(Q3 - Q2), |A\ x\ Q2|]$$

In most cases, these distances are just the difference between the median and the first and third quartile of the E values respectively. However, in some unusual distributions it is possible for the data to group around the median leading to businesses being erroneously identified as suspicious. To safeguard against this happening, a small multiple A of the median of the E values is used instead. The parameter A is commonly given the value 0.05 for this purpose. Suspicious businesses are then identified as follows:

$$if$$

$$E < Q2 - C\ x\ d_{Q1}$$

$$or$$

$$E < Q2 - C\ x\ d_{Q3}$$

Then treat as suspicious

IV. Methods for Correcting Errors in VAT Data

These methods are considered for each type of error identified above, and tested to identify which ones work best for VAT data. Once an error or suspicious value has been identified in VAT data, there are a number of options for dealing with it. The main options are described below:

a) **Ignore the potential error.** This is probably the least useful choice. However, depending on the use of the data it may be considered that some values will not have a significant impact on statistical outputs whether they are in error or not.

b) **Remove any suspicious values from the data set**. The implication is that only nonsuspicious values are used for statistical outputs. This can be a good solution when it is not necessarily required to use VAT data for every unit in the study population. For example, when other sources of information on Turnover are available.

c) **Mark the value as suspicious, but do not change it.** This allows any user of the data to make their own choice of how to deal with suspicious values. This allows for a great deal of flexibility for different statistical uses of VAT data. However, for users with insufficient understanding of the consequences of their choice or of the options available, this could lead to inaccurate outputs.

d) **Change suspicious values manually.** Any value deemed suspicious is inspected by a statistician and changed to be more in line with expectations. This can lead to more accurate data, but the methods for changing values may be inconsistent and subject to the perceptions of the statistician. This is also a resource intensive option when there are a significant number of suspicious values to change, especially for large VAT datasets. A variation on this option would be to only manually review a subset of suspicious values, thought to be of particular importance (due to their size or expected impact on statistical outputs).

e) **Change suspicious values automatically.** Imputation methods are used to automatically alter suspicious values to be more in line with expectations. Once set up, this option requires no resource. However, since values are changed automatically it is important to have a well specified method to avoid causing large errors in the data set.

f) **Contact suspicious businesses to confirm their values.** Contacting businesses can be resource intensive and can place extra burden on businesses. This can be reduced by only contacting the most important businesses (or those with the most suspicious values) and dealing with the rest differently. This might be expected to be the most accurate option for those businesses that are contacted, although this assumes that businesses are able to give the correct figures at the time of contact.

When choosing an option for dealing with errors in VAT data it is important to consider the statistical uses of the data, the resource available for cleaning the data and the impact of any possible additional burden on businesses. When the value is changed, it is advisable to keep a

record of the original value. This assists with future analysis of the method for changing the value and also ensures that the original data set can be re-created whenever required. In many cases, the most sensible and cost-effective option will be to automatically change suspicious values using imputation methods. A range of imputation methods are discussed and tested below.

1. Methods for correcting systematic errors
It is quite straightforward to correct systematic errors by simply multiply the suspicious values by 1000 (or divide the suspicious value by 1000, if the error is in the opposite sense). Since we do not have the possibility to re-contact businesses to check their VAT return, we need some independent sources to check if we have identified the errors correctly. As far as Turnover is concerned, we can compare the VAT Turnover of a business against the value of its Turnover in the Business Register (BR).

2. Methods for correcting suspicious data patterns
The five types of pattern discussed in the chapter *Methods for detecting suspicious data patterns* can be grouped into 3 categories as below. Methods for dealing with each of these groups are considered separately:

Type 1

- Zero VAT data values in three quarters, positive VAT data value in the other quarter.
- Same VAT data values in all four quarters.
- Same VAT data values for three quarters, a different (positive) VAT data value in the other quarter.

Type 2

- Patterns that suggest the business has failed to report VAT data in one quarter

Type 3

- Negative VAT data

<u>Methods for dealing with type 1</u>

For the first group of the 3 patterns, it is likely that the single positive VAT data figure is an annual figure. In the second pattern, it appears that the annual VAT data has been split evenly into the quarters. In the third pattern, it is likely that the same estimated figure has been given for three of the quarters and the fourth quarter has been used as a balancing item to give the correct overall VAT data for the year.

In each case, if our assumptions are correct, we can deduce a reliable annual VAT data figure (by summing the values from the four quarters). However, we have no information on how the annual

Turnover should be split into quarterly VAT data figures. To create usable data, it is necessary to impute quarterly VAT data figures. Given that we have annual VAT data, the obvious method is to impute the proportions of VAT data for each quarter and multiply those proportions by the annual figure.

Three methods are tested for imputing the quarterly VAT data proportions, based on either the proportions of quarterly VAT data value for similar businesses or the proportions for the same business in the previous year. The three methods are as follows:

- Mean proportions within the homogenous class (in the same year)
- Median proportions within the homogeneous class (in the same year)
- Proportions for the same business in the previous year

The mean and median proportions are calculated using all businesses in the homogeneous class with acceptable VAT data values in every quarter. Note that the mean and median proportions will not necessarily add to one. It will be necessary to re-scale the imputed quarterly VAT data values to add up to the reported annual VAT data figure. This is achieved by multiplying the imputed quarterly values by the reported annual figure and dividing by the sum of the imputed quarterly values.

Methods for dealing with type 2

These patterns suggest that the business has failed to report VAT data in one quarter. This category includes only one suspicious pattern: Zero VAT data value in one quarter, positive VAT data values in the other three quarters.

In most cases, it can be assumed that this pattern results from the business not giving their true VAT data figure for the one quarter with a zero value. Note that there may be some exceptional industries where this pattern is expected.

Since this pattern implies an unusable value for one figure, the imputation methods available are the same as those discussed below for suspicious VAT data values in a single period.

Method for dealing with type 3

This covers the remaining suspicious patterns: Negative VAT data value in any of the quarters. The method of dealing with negative VAT data may depend on the reason for the occurrence of negative VAT data value. However, it is generally not advised to simply reverse the sign and treat the value as being positive. The recommended solution is to treat the value as a single suspicious value and use one of the imputation methods described below to replace it.

3. Methods for correcting random errors

Correcting random error is a well-known problem in business surveys and over the years, a range of imputation methods have been developed that can be reused for our problem of correcting

suspicious values in VAT data. However, there is no guidance on which imputation methods work well with administrative sources. Ten imputation methods for suspicious VAT data values in a single period are described below. These take account of methods currently used for editing VAT data as well as covering the main imputation methods used in business surveys.

Method 1 - Mean imputation

This method simply replaces a suspicious VAT data value with the mean of the VAT data values of all businesses that are not suspicious. To improve the accuracy of the method, it is advisable to split the VAT data set into homogeneous classes and calculate means within those classes. Businesses with suspicious values are then changed to their class mean. Variables such as NACE industry and Employment size are often used to create the homogeneous classes.

Method 2 - Trimmed mean imputation

Because means are influenced by outliers it is often sensible to trim (i.e. remove) extreme values before calculating the mean. Method 2 is the trimmed mean, where the highest and lowest 10% of VAT data values are removed from each homogeneous class before taking the mean.

Method 3 - Median imputation

This method simply replaces a suspicious VAT data value with the median of the VAT data values of all businesses that are not suspicious. The median is more robust than the mean. As with mean imputation, the method usually gives better results if medians are calculated within homogeneous classes. Because the median is robust, there is no need to trim extreme values.

Method 4 - Ratio imputation: ratio of means using previous period

The method involves using a previous value for the business and multiplying it by a ratio based on the growth between the previous and current period for similar businesses.

Imputed value = Previous value for business x Ratio

$$Ratio = \frac{\sum_{i \in class} current\ period\ value\ for\ business\ i}{\sum_{i \in class} previous\ period\ value\ for\ business\ i}$$

The businesses whose current and previous VAT data values are included in the means are all of those with acceptable values in both periods and which belong to the same homogeneous class as the business being imputed. Note that exactly the same businesses contribute to the two means. This is why the ratio of means reduces to a ratio of sums.

Method 5 - Ratio imputation: ratio of means using same period in previous year

This method involves computing the ratio imputation using the ratio of means with the growth calculated from the same period in the previous year.

Method 6 - Ratio imputation: mean of ratios using previous period

This method involves computing the ratio imputation using the mean of the ratios from the previous period and is defined as follows:

$$imputed\ value = \ R\ x\ previous\ value\ for\ suspicious\ business$$

$$where\ R = \frac{1}{number\ of\ businesses\ in\ class}\ x \sum_{i \in class} \frac{current\ period\ value\ for\ business\ i}{previous\ period\ value\ for\ business\ i}$$

Note that the businesses whose ratios are summed are all of those with acceptable values in both periods and who belong to the same homogeneous class as the business being imputed.

Method 7 - Ratio imputation: trimmed mean of ratios using previous period

The average ratio can be unduly influenced by businesses with unusually large or unusually small ratios. For this reason, the extreme ratios are often trimmed before calculating the mean. This method involves computing the mean of ratios based on the previous period with the highest and lowest 10% of ratios in each homogeneous class trimmed before taking the mean.

Method 8 - Ratio imputation: mean of ratios using same period in previous year

For completeness, we also consider the mean of ratios imputation using the same period in the previous year, instead of the previous period. This method is based on ratio imputation using the (untrimmed) mean of ratios based on the previous year.

Method 9 - Ratio imputation: trimmed mean of ratios using same period in previous year

The highest and lowest 10% of ratios in each homogeneous class are trimmed before calculating the means.

Method 10 - Donor imputation

This method involves using the VAT data from a donor business in place of the suspicious value. The donor is the (non-suspicious) business within the homogeneous class that is most similar to the business whose VAT data value needs imputing. Similar businesses are identified by calculating distances between the suspicious business and all potential donors. Distances are calculated separately for specified auxiliary variables, known as matching variables. In the evaluation, Turnover and Employment from the business register were used as matching variables. The distances for each matching variable are aggregated together and the potential donor with the smallest combined distance is chosen as the donor.

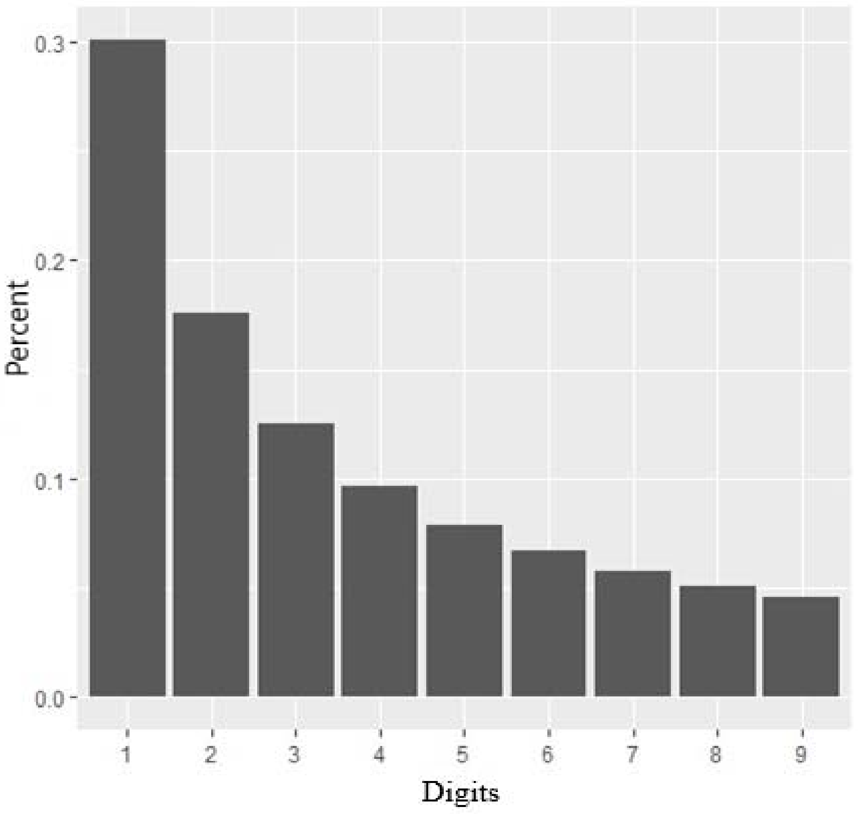
V. Data Quality Mining

Data mining helps discover specific data patterns in large data sets. A technique that fits in this category is checking the conformity of data to Benford's Law. It is a simple technique and a good example of a very nonintuitive pattern. The law pertains to the first digits of a collection of numbers, like stock market prices or number of inhabitants of cities. Against most people's intuition, the "1" digit leads approximately 30% of the time, going down gradually with "9" occurring less than 5% of the time. The distribution of first digits can be calculated with the following formula:

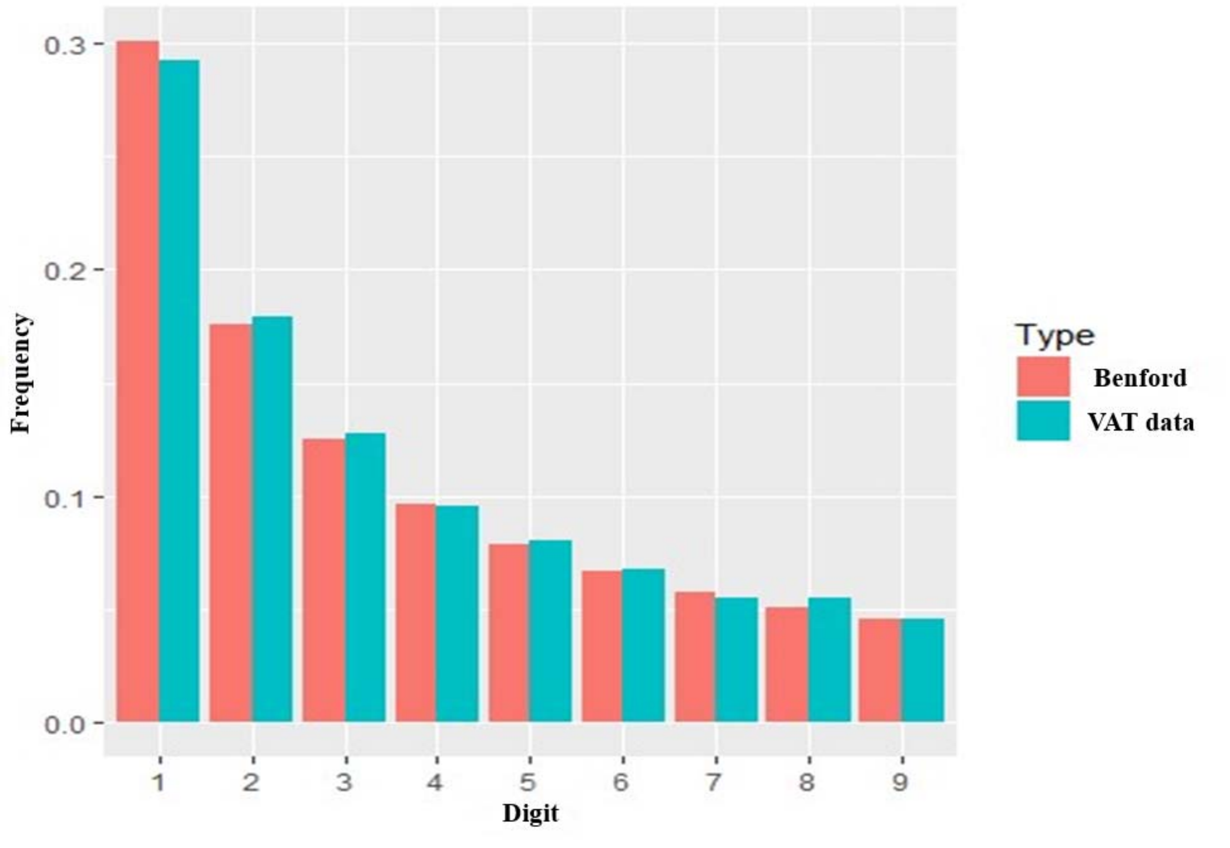
$$P(d) = \log10\ [(1 + d) / d],\ d = 1{:}9$$

Graphic illustration - the distribution of the first digit:

Digits	Benford's Law
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%



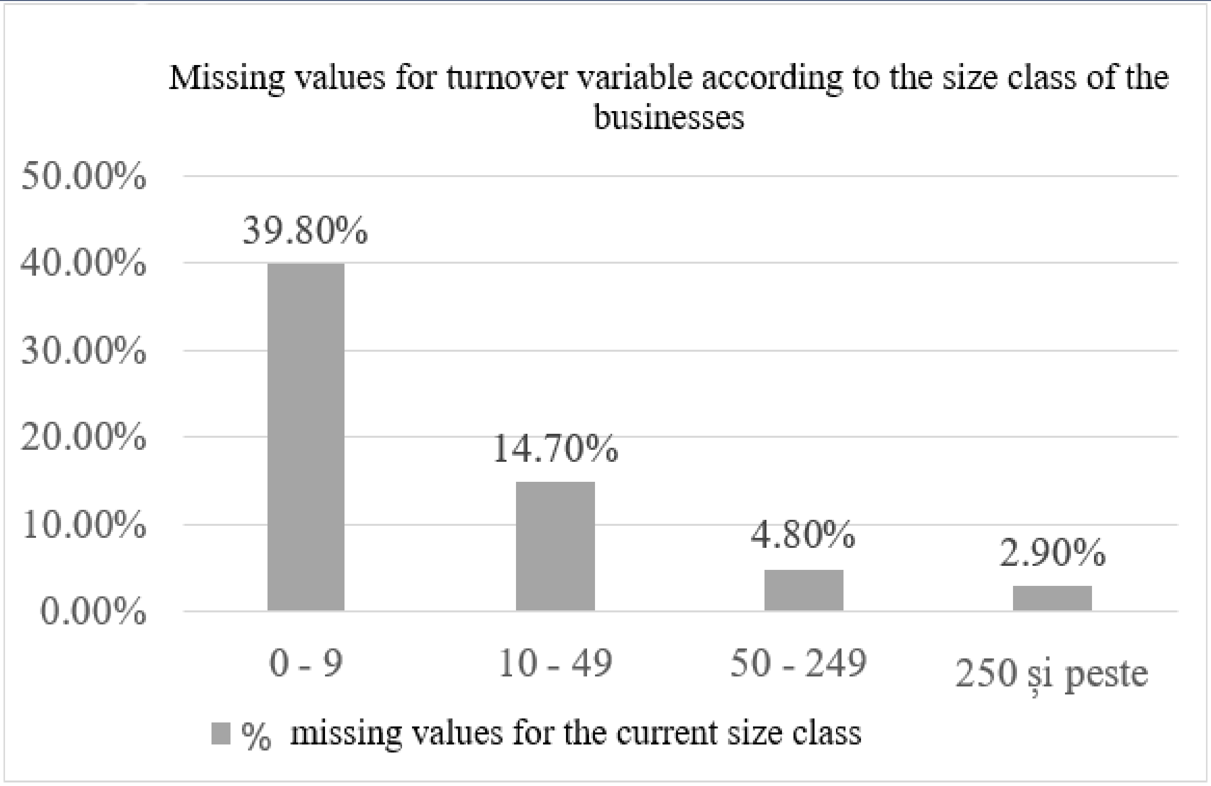
The following chart shows the distribution of the first digit for a VAT data sample. The sample contains the turnover variable for businesses with more than 250 employees. The graph shows that VAT data follow Benford's Law closely.

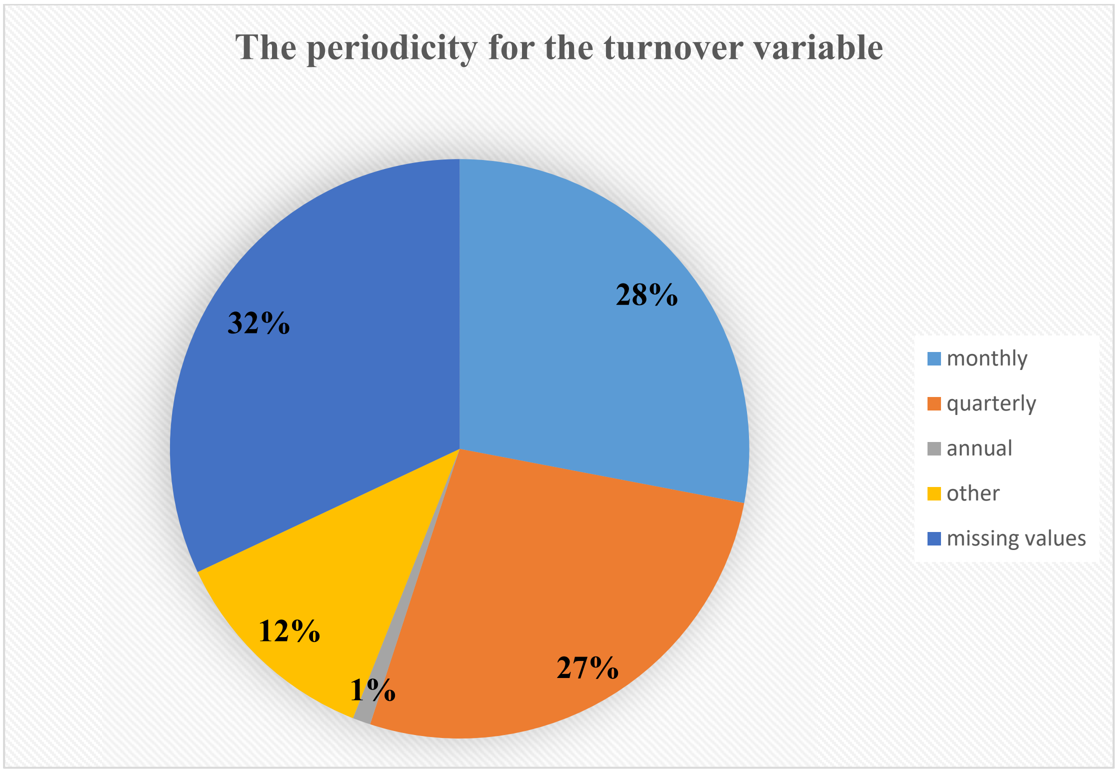


VI. Annexes

1. The quality indicators computed on VAT data by applying the above methods for the year 2016.

Indicator	Percent
Misclassification rate	24%
Undercoverage	13%
Overcoverage	12%
Size of revisions from the different versions of the admin data RAR – Relative Absolute Revision	49%


Missing values for turnover variable according to the size class of the businesses
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
39.80%
14.70%
4.80%
2.90%
0 - 9
10 - 49
50 - 249
250 și peste
% missing values for the current size class


The periodicity for the turnover variable
28%
27%
1%
12%
32%
monthly
quarterly
annual
other
missing values

2. The R code used to compute the quality indicators and to develop the methods for detecting errors in VAT dataset

Loading libraries

```r
library(dplyr)
```

Indicator – Missing values

```r
indicator_missing = function(sursa){
  sursa = read.csv(sursa)
  indicator = list()
  for (i in 1:4){
    val_lipsa = subset(sursa, t_r1_r == 0 & t_r2_r  == 0 & t_r3_r  == 0 &
                              t_r4_r == 0 & t_r5_r  == 0 & t_r6_r  == 0 &
                              t_r7_r == 0 & t_r8_r  == 0 & t_r9_r  == 0 &
                              t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 0 &
                              cls_marime == i )
    val_total = subset(sursa, cls_marime == i)
    indicator[i] = nrow(val_lipsa)/nrow(val_total)*100
  }
  indicator_missing = data.frame(cls1 = c(indicator[[1]]),
                                 cls2 = c(indicator[[2]]),
                                 cls3 = c(indicator[[3]]),
                                 cls4 = c(indicator[[4]]))
  return(indicator_missing)
}
```

Example: Applying the function indicator_missing for the VAT dataset, the year 2016.

```r
indicator_missing("sursa2016rev.csv")

##      cls1    cls2     cls3     cls4
## 1 39.8503 14.7601 4.867971 2.979666
```

Indicator - Misclassification rate

```r
indicator_caen = function(sursa, REGIS){
  sursa = read.csv(sursa)
  REGIS = read.csv(REGIS)
  sursa_merged = inner_join(sursa, REGIS, by = c("cod" = "codi"))
  indicator_caen = nrow(subset(sursa_merged, caen != 0 & caen != caen_rev2))/
                   nrow(subset(sursa_merged, caen != 0)) * 100
  return(indicator_caen)
}
```

Example: Applying the function indicator_caen for the VAT dataset, the year 2016.

```
indicator_caen("sursa2016rev.csv", "iactiv15.csv")

## [1] 4.856126
```

Indicator - Undercoverage

```
indicator_undercoverage = function(sursa, REGIS){
  sursa = read.csv(sursa)
  REGIS = read.csv(REGIS)
  indicator_undercoverage = nrow(subset(REGIS,
                            !(REGIS$codi %in% as.list(sursa$cod))))/
                            nrow(REGIS) * 100
  return(indicator_undercoverage)
}
```

Example: Applying the function indicator_undercoverage for the VAT dataset, the year 2016.

```
indicator_undercoverage("sursa2016rev.csv", "iactiv15.csv")

## [1] 13.13882
```

Indicator - Overcoverage

```
indicator_overcoverage = function(sursa, REGIS){
  sursa = read.csv(sursa)
  REGIS = read.csv(REGIS)
  indicator_overcoverage = nrow(subset(sursa,
                           !(sursa$cod %in% as.list(REGIS$codi))))/
                           nrow(REGIS) * 100
  return(indicator_overcoverage)
}
```

Example: Applying the function indicator_overcoverage for the VAT dataset, the year 2016.

```
indicator_overcoverage("sursa2016rev.csv", "iactiv15.csv")

## [1] 13.23506
```

Indicator - RAR – Relative Absolute Revision

```
indicator_rar = function(sursa, sursa_anterioara ){
  sursa = read.csv(sursa)
  sursa_anterioara = read.csv(sursa_anterioara)
  m = inner_join(sursa, sursa_anterioara[,c("cod", "ca_01", "ca_02", "ca_03",
      "ca_04", "ca_05", "ca_06", "ca_07", "ca_08", "ca_09", "ca_10", "ca_11",
"ca_12")],
      by = c("cod"))
  indicator_rar = (abs(sum(m$ca_01_r) - sum(m$ca_01)) + abs(sum(m$ca_02_r) -
sum(m$ca_02))
```

```
                  + abs(sum(m$ca_03_r) - sum(m$ca_03)) + abs(sum(m$ca_04_r) -
sum(m$ca_04))
                  + abs(sum(m$ca_05_r) - sum(m$ca_05)) + abs(sum(m$ca_06_r) -
sum(m$ca_06))
                  + abs(sum(m$ca_07_r) - sum(m$ca_07)) + abs(sum(m$ca_08_r) -
sum(m$ca_08))
                  + abs(sum(m$ca_09_r) - sum(m$ca_09)) + abs(sum(m$ca_10_r) -
sum(m$ca_10))
                  + abs(sum(m$ca_11_r) - sum(m$ca_11)))/
                      (sum(m$ca_01) + sum(m$ca_02) + sum(m$ca_03) + sum(m$ca_0
4)
                  +  sum(m$ca_05) + sum(m$ca_06) + sum(m$ca_07) + sum(m$ca_08)
                  +  sum(m$ca_09) + sum(m$ca_10) + sum(m$ca_11) ) * 100
  return(indicator_rar)
}
```

Example: Applying the function indicator_rar for the VAT dataset, the year 2016.

```
indicator_rar("sursa2016rev.csv", "sursa2016.csv")

## [1] 49.32423
```

The Periodicity for turnover variable

Loading libraries

```
library(dplyr)
```

Metoda

```
periodicitate = function(sursa){
  sursa = read.csv(sursa)
  periodicitate <- sursa %>% group_by(t_r1_r, t_r2_r, t_r3_r, t_r4_r, t_r5_r,
                             t_r6_r, t_r7_r, t_r8_r, t_r9_r, t_r10_r,
                             t_r11_r, t_r12_r) %>% dplyr :: summarise(n=n())
  periodicitate <- as.data.frame(arrange(periodicitate, desc(n)))
  return(periodicitate[1:6,])
}
# Writing the file "periodicitate"
## write.csv(periodicitate, "path/periodicitate.csv" )
```

Example: Applying the function periodicitate for the VAT dataset, the year 2016.

```
periodicitate("sursa2016rev.csv")

## Warning: package 'bindrcpp' was built under R version 3.4.4

##   t_r1_r t_r2_r t_r3_r t_r4_r t_r5_r t_r6_r t_r7_r t_r8_r t_r9_r t_r10_r
## 1      0      0      0      0      0      0      0      0      0       0
## 2      1      1      1      1      1      1      1      1      1       1
## 3      0      0      1      0      0      1      0      0      1       0
## 4      0      0      0      0      0      0      0      0      0       0
## 5      0      0      1      0      0      1      0      0      1       0
## 6      0      0      1      0      0      0      0      0      0       0
##   t_r11_r t_r12_r      n
## 1       0       0 165157
## 2       1       1 147983
## 3       0       1 140788
## 4       0       1   5752
## 5       0       0   5752
## 6       0       0   4556
```

Detecting systematic errors

Loading libraries

```
library(dplyr)

calculeazaEroriUnitateDeMasura = function(sursa_an_curent, sursa_an_precedent
){
  #Selecting the parameters
  n = 12 #number of months
  A = 0.00065
  B = 0.00135

  sursa_an_curent = read.csv(sursa_an_curent)
  sursa_an_precedent = read.csv(sursa_an_precedent)

  colnames(sursa_an_precedent)[which(names(sursa_an_precedent) == "ca_12_r")]
= "ca_12_rr"

  # Join between sursa_an_curent and sursa_an_precedent:
  sursa_an_curent = left_join(sursa_an_curent, sursa_an_precedent[,c("cod","c
a_12_rr")],
                             by = c("cod"))

  for(i in 2:n){
    col1 = paste0("Metoda0_",i)
    col2 = paste0("t_r",i, "_r")
    col3 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")
    col4 = paste0(ifelse(i-1 <= 9, "ca_0", "ca_"), i-1, "_r")

    sursa_an_curent[,"Metoda0_1"] = ifelse(sursa_an_curent[,"t_r1_r"] == 1 &
                                      sursa_an_curent[,"ca_12_rr"]!= 0 &
                                      sursa_an_curent[,"ca_01_r"]/
                                        sursa_an_curent[,"ca_12_rr"] > A &
                                      sursa_an_curent[,"ca_01_r"]/
                                        sursa_an_curent[,"ca_12_rr"] < B,
                                      1, 0)

    sursa_an_curent$Metoda0_1[is.na(sursa_an_curent$Metoda0_1)] = 0

    sursa_an_curent[,col1] = ifelse(sursa_an_curent[,col2] == 1 &
                                sursa_an_curent[,col4] != 0 &
                                sursa_an_curent[,col3]/sursa_an_curent[,col4]
> A &
                                sursa_an_curent[,col3]/sursa_an_curent[,col4]
< B , 1, 0)
  }

  # The number of outliers for every month
  p = colSums(sursa_an_curent %>% select(Metoda0_1:Metoda0_12), na.rm = TRUE)

  # Percent of outliers for every month
  procent = vector("numeric", 12L)
  for (i in 1:n){
    col = paste0("t_r", i, "_r")
    procent[i] = paste0( round(p[[i]]*100/sum(sursa_an_curent[,col]),2), "% "
)
  }

  luna = c("ianuarie", "februarie", "martie", "aprilie", "mai", "iunie", "iul
ie",
           "august", "septembrie", "octombrie", "noiembrie", "decembrie")
  procent = data.frame(luna, procent)
  return(procent)

  # Writing the file "sursa_an_curent"
  # 1 - The turnover is suspicious
  # 0 - The turnover is not suspicious
  ## write.csv(sursa_an_curent, "path/sursa_metoda0.csv")
}
```

Example: Applying the function calculeazaEroriUnitateDeMasura() for the VAT dataset, the year 2016.

```
  calculeazaEroriUnitateDeMasura("sursa2016rev.csv", "sursa2015rev.csv")

##           luna procent
## 1     ianuarie   0.15%
## 2    februarie   0.01%
## 3       martie   0.01%
## 4      aprilie   0.02%
## 5          mai   0.02%
## 6        iunie   0.01%
## 7        iulie   0.02%
## 8       august   0.03%
## 9   septembrie   0.01%
## 10   octombrie   0.03%
## 11   noiembrie   0.01%
## 12   decembrie   0.01%
```

Suspicious data patterns

Loading libraries

```
library(dplyr)

calculeazaPatternValoriAberante = function(sursa){
  sursa = read.csv(sursa)
  p1 = subset(sursa, ca_03_r == 0 & ca_06_r == 0 & ca_09_r == 0 & ca_12_r > 0
&
              t_r1_r == 0 &  t_r2_r == 0 &  t_r3_r == 1 &
              t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t_r8_
r == 0 &
              t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)
  procent1 = paste0(round(nrow(p1)/nrow(subset(sursa,  t_r1_r == 0 &  t_r2_r
== 0 &
    t_r3_r == 1 & t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t
_r8_r == 0 &
    t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)) * 100, 2), "%
")

  p2 = subset(sursa, ca_03_r == 0 & ca_06_r > 0 & ca_09_r > 0 & ca_12_r > 0 &
       t_r1_r == 0 &  t_r2_r == 0 &  t_r3_r == 1 &
       t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 &
       t_r8_r == 0 &  t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r ==
1)
  procent2 = paste0(round(nrow(p2)/nrow(subset(sursa,  t_r1_r == 0 &  t_r2_r
== 0 &
    t_r3_r == 1 & t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t
_r8_r == 0 &
    t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)) * 100, 2), "%
")

  p3 = subset(sursa, ca_03_r == ca_06_r & ca_06_r == ca_09_r & ca_09_r == ca_
12_r &
                t_r1_r == 0 &  t_r2_r == 0 &  t_r3_r == 1 &
                t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t_r
8_r == 0 &
                t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)
  procent3 = paste0(round(nrow(p3)/nrow(subset(sursa,  t_r1_r == 0 &  t_r2_r
== 0 &
    t_r3_r == 1 & t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t
_r8_r == 0 &
    t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)) * 100, 2), "%
")

  p4 = subset(sursa, ca_03_r == ca_06_r & ca_06_r == ca_09_r & ca_12_r > 0 &
                ca_12_r != ca_09_r &
                t_r1_r == 0 &  t_r2_r == 0 &  t_r3_r == 1 &
                t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t_r
8_r == 0 &
                t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)
  procent4 = paste0(round(nrow(p4)/nrow(subset(sursa,  t_r1_r == 0 &  t_r2_r
== 0 &
    t_r3_r == 1 & t_r4_r == 0 &  t_r5_r == 0 &  t_r6_r == 1 & t_r7_r == 0 & t
_r8_r == 0 &
    t_r9_r == 1 & t_r10_r == 0 & t_r11_r == 0 & t_r12_r == 1)) * 100, 2), "%
")


  pattern = cat(
  "Procentul unitatilor care au depus cifra de afaceri 0 in primele trei trim
estre
   si cifra de afaceri pozitiva in trimestrul patru este:",procent1, "\n",
  "Procentul unitatilor care au depus cifra de afaceri 0 in primul trimestru
si valori
   pozitive ale cifrei de afaceri in celelalte trei trimestre este:", procent
2, "\n",
  "Procentul unitatilor care au depus aceeasi cifra de afaceri in toate cele
patru
   trimestre este:",procent3,"\n",
  "Procentul unitatilor au depus aceeasi cifra de afaceri pe trei trimestre s
i o alta
   valoare pozitiva in trimestrul patru este:", procent4)
}
```

Example: Applying the function calculeazaPatternValoriAberante() for the VAT dataset, the year 2016.

```
calculeazaPatternValoriAberante("sursa2016rev.csv")

## Procentul unitatilor care au depus cifra de afaceri 0 in primele trei trim
estre
##    si cifra de afaceri pozitiva in trimestrul patru este: 2.09%
##  Procentul unitatilor care au depus cifra de afaceri 0 in primul trimestru
si valori
##    pozitive ale cifrei de afaceri in celelalte trei trimestre este: 4.37%
##  Procentul unitatilor care au depus aceeasi cifra de afaceri in toate cele
patru
##    trimestre este: 11.65%
##  Procentul unitatilor au depus aceeasi cifra de afaceri pe trei trimestre
si o alta
##    valoare pozitiva in trimestrul patru este: 2.25%
```

Quartile1, Median, Quartile3 according to NACE, size class and periodicity

Loading libraries

```
library(dplyr)
library(lazyeval)

calculeazaQuartileMediana = function(sursa){
  sursa = read.csv(sursa)
  #Selecting the parameters
  n = 12 #number of months

  # Defining an empty dataframe
  randuri = read.csv(text="div_caen, cls_marime, periodicit, q1, m, q3, luna"
)

  for(clasa_marime in 1:4) {
    for (i in 1:n) {
      coloana_ca = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r");
      conditie_t_r = interp(~ col_tr == 1, col_tr = as.name(paste0("t_r", i,
"_r")));
      un_rand <- sursa %>% filter_(~cls_marime == clasa_marime,
                                  ~periodicit == 1, conditie_t_r,
                                  ~div_caen != 0 ) %>% group_by(div_caen
) %>%
      summarise_(cls_marime = clasa_marime, periodicit =1,
           q1 = interp(~quantile(col_ca, probs = 0.25),
           col_ca = as.name(coloana_ca)), m = interp(~median(col_ca),
           col_ca = as.name(coloana_ca)), q3 = interp(~quantile(col_ca, probs
= 0.75),
           col_ca = as.name(coloana_ca)), luna = i);
      randuri = bind_rows(randuri, un_rand);    }
  }

  for(clasa_marime in 1:4) {
    for (i in c(3,6,9,12)) {
      coloana_ca = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r");
      conditie_t_r = interp(~ col_tr == 1, col_tr = as.name(paste0("t_r", i,
"_r")));
      un_rand <- sursa %>% filter_(~cls_marime == clasa_marime, ~periodicit =
= 2,
                                  conditie_t_r, ~div_caen != 0 ) %>%
      group_by(div_caen) %>% summarise_(cls_marime = clasa_marime, periodicit
=2,
      q1 = interp(~quantile(col_ca, probs = 0.25), col_ca = as.name(coloana_c
a)),
      m = interp(~median(col_ca), col_ca = as.name(coloana_ca)),
      q3 = interp(~quantile(col_ca, probs = 0.75), col_ca = as.name(coloana_c
a)),
           luna = i);
      randuri = bind_rows(randuri, un_rand);    }
  }
  return(head(randuri))

  # Writing the file "randuri"
  ## write.csv(randuri, "path/q1_m_q3.csv")
}
```

Example: Applying the function calculeazaQuartileMediana() for the VAT dataset, the year 2016.

```
calculeazaQuartileMediana("sursa2016rev.csv")

## Warning: package 'bindrcpp' was built under R version 3.4.4

##   div_caen cls_marime periodicit     q1      m        q3 luna
## 1        1          1          1      0      0   9986.50    1
## 2        2          1          1      0   9582  46029.25    1
## 3        3          1          1      0   2156  25048.00    1
## 4        5          1          1 325020 325020 325020.00    1
## 5        6          1          1      0      0      0.00    1
## 6        7          1          1      0      0   8543.50    1
```

The Method Quartile distances

Loading libraries

```
library(tidyverse)
library(dplyr)
library(reshape2)
library(lazyeval)
library(stringi)
library(stringr)

calculeazaMetodaIntervalulInterquartilic = function(sursa, quartile_mediana){
  sursa = read.csv(sursa)
  quartile_mediana = read.csv(quartile_mediana)

  n = 12
  c = 10

  model1 = function(data) {

    ndata = names(data)
    n = sum(str_detect(ndata, 'LB'))

    for (i in 1:n){
      col1 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")
      col2 = paste0('LB_', i)
      col3 = paste0('UB_', i)

      new_col_name = paste0('Metoda1_', i)

      mutate_call = lazyeval::interp(~ ifelse(a < b | a > c, 1, 0), a = as.na
me(col1),
                                     b = as.name(col2),
                                     c = as.name(col3))
      data = data %>% mutate_(.dots = setNames(list(mutate_call), new_col_nam
e))
    }

    return(data)
  }


  sursa$div_caen[is.na(sursa$div_caen)] = 0

  quartile_mediana2 = quartile_mediana %>% mutate(LB = q1 - c*(m-q1),
                                                  UB = q3+c*(q3-m))
  quartile_mediana2 = quartile_mediana2[, !names(quartile_mediana2)%in%c("q1"
, "m", "q3")]

  quartile_mediana_melt2 = melt(quartile_mediana2[,-1], id.vars = c("div_caen
",
                                                "cls_marime", "periodicit", "luna
"))
  quartile_mediana_cast = dcast(quartile_mediana_melt2, div_caen + cls_marime
+
                                periodicit ~ variable+luna)

  sursa_merged_lj = left_join(sursa, quartile_mediana_cast, by = c("div_caen"
,
                                                                   "cls_marim
e",
                                                                   "periodici
t"))

  sursa_merged_lj_mutate = model1(sursa_merged_lj)

  sursa_merged_lj_mutate = sursa_merged_lj_mutate[, !names(sursa_merged_lj_mu
tate)%in%
                           c("LB_1", "LB_2", "LB_3", "LB_4", "LB_5", "LB_6",
"LB_7",
                             "LB_8", "LB_9", "LB_10", "LB_11", "LB_12", "UB_1
",
                             "UB_2", "UB_3", "UB_4", "UB_5", "UB_6", "UB_7",
                             "UB_8", "UB_9", "UB_10", "UB_11", "UB_12")]


  sursa_merged_lj_mutate[c("Metoda1_1", "Metoda1_2", "Metoda1_3", "Metoda1_4"
,
                           "Metoda1_5", "Metoda1_6", "Metoda1_7", "Metoda1_8"
,
                           "Metoda1_9", "Metoda1_10", "Metoda1_11",
                           "Metoda1_12")][is.na(sursa_merged_lj_mutate[c("Met
oda1_1",
                           "Metoda1_2", "Metoda1_3", "Metoda1_4", "Metoda1_5"
,
                           "Metoda1_6", "Metoda1_7", "Metoda1_8", "Metoda1_9"
,
                           "Metoda1_10","Metoda1_11","Metoda1_12")])] <- 0


  p = colSums(sursa_merged_lj_mutate %>% select(Metoda1_1:Metoda1_12), na.rm
= TRUE)
  p
  procent <- vector("numeric", 12L)
  for (i in 1:12){
    col = paste0("t_r", i, "_r")
    procent[i] <- paste0( round(p[[i]]*100/sum(sursa[,col]),2), "%")
  }

  luna = c("ianuarie", "februarie", "martie", "aprilie", "mai", "iunie", "iul
ie",
              "august", "septembrie", "octombrie", "noiembrie", "decembrie")
  procent = data.frame(luna, procent)
  return(procent)


  ## write.csv(sursa_merged_lj_mutate, path/sursa_metoda1.csv")
}
```

Example: Applying the function calculeazaMetodaIntervalulInterquartilic() pentru sursa din anul 2016

```
calculeazaMetodaIntervalulInterquartilic("sursa2016rev.csv", "q1_m_q3.csv")

## Warning: package 'bindrcpp' was built under R version 3.4.4

##          luna procent
## 1    ianuarie   2.11%
## 2   februarie   2.05%
## 3      martie   1.22%
## 4     aprilie   1.94%
## 5         mai   1.94%
## 6       iunie   1.19%
## 7       iulie   1.91%
## 8      august   1.88%
## 9  septembrie   1.26%
## 10  octombrie    1.9%
## 11  noiembrie   1.98%
## 12  decembrie   1.37%
```

The Method Period on period ratios

Loading libraries

```
library(tidyverse)
library(plyr)
library(reshape2)
library(lazyeval)
library(stringi)
library(stringr)
library(dplyr)
library(data.table)

calculeazaMetodaRaportPerioada = function(sursa_an_curent, sursa_an_precedent
,
                                quartileMediana_curent, quartileMediana_prece
dent){
  sursa_an_curent = read.csv(sursa_an_curent)
  sursa_an_precedent = read.csv(sursa_an_precedent)
  quartileMediana_curent = read.csv(quartileMediana_curent)
  quartileMediana_precedent = read.csv(quartileMediana_precedent)


  n = 12

  sursa_an_curent$div_caen[is.na(sursa_an_curent$div_caen)] = 0
  setnames(sursa_an_precedent, "ca_12_r", "ca_12_rr" )



  sursa_an_curent = left_join(sursa_an_curent, sursa_an_precedent[,c("cod","c
a_12_rr")],
                              by = c("cod"))


  sursa_m15 <-  quartileMediana_precedent[,c(2,3,4,6,8)] %>% filter(luna == 1
2)

  setnames(sursa_m15, "m", "m15" )

  for(i in 1:n){
    col1 = assign(paste0("sursa_m",i),
                quartileMediana_curent[,c(2,3,4,6,8)] %>% filter(luna == i)
)
    setnames(col1, "m", paste0("m",i) )
  }
  sursa_merged = join_all(list(sursa_an_curent, sursa_m1, sursa_m2, sursa_m3,
sursa_m4,
                              sursa_m5, sursa_m6, sursa_m7,   sursa_m8, surs
a_m9,
                              sursa_m10, sursa_m11, sursa_m12, sursa_m15),
                              by = c("div_caen", "cls_marime", "periodicit")
,
                              type='left')
  sursa_merged = sursa_merged[, !names(sursa_merged)%in%c("luna")]


  sursa_merged[,"score15"] <- sursa_merged[,"ca_12_rr"]/sursa_merged[,"m15"]
  for (i in 1:n){
    col1 = paste0("score",i)
    col2 = paste0("m",i)
    col3 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")
    sursa_merged[,col1] <- sursa_merged[,col3]/sursa_merged[,col2]
  }


  sursa_merged[,"score15"][is.nan(sursa_merged[,"score15"])] <- NA
  sursa_merged[,"score15"][is.infinite(sursa_merged[,"score15"])] <- NA
  for(i in 1:n){
    col = paste0('score',i)
    sursa_merged[,col][is.nan(sursa_merged[,col])] <- NA
    sursa_merged[,col][is.infinite(sursa_merged[,col])] <- NA
  }



  sursa_merged[,"test1"] <- ifelse(sursa_merged[,"score1"] > sursa_merged[,"s
core15"],
                                  sursa_merged[,"score1"] / sursa_merged[,"s
core15"],
                                  sursa_merged[,"score15"] / sursa_merged[,"
score1"])
  for(i in 1:(n-1)){
    col1 = paste0("test",i+1)
    col2 = paste0("score", i+1)
    col3 = paste0("score", i)
    sursa_merged[,col1]<- ifelse(sursa_merged[,col2] > sursa_merged[,col3],
                             sursa_merged[,col2] / sursa_merged[,col3],
                             sursa_merged[,col3] / sursa_merged[,col2])
   }


  for(i in 1:n){
    col = paste0('test',i)
    sursa_merged[,col][is.nan(sursa_merged[,col])] <- NA
    sursa_merged[,col][is.infinite(sursa_merged[,col])] <- NA
  }


  for(i in 1:n){
    col1 = paste0("Metoda2_",i)
    col2 = paste0("test",i)
    sursa_merged[,col1] <- ifelse(!is.na(sursa_merged[,col2]) &
                                  sursa_merged[,col2] > 30 , 1, 0)
  }


  sursa_merged = sursa_merged[, !names(sursa_merged)%in%c("m1", "m2",
                 "m3", "m4", "m5", "m6", "m7", "m8", "m9", "m10", "m11",
                 "m12", "m15", "score1", "score2", "score3", "score4",
                 "score5", "score6","score7", "score8", "score9","score10",
                 "score11","score12","score15", "test1", "test2", "test3",
                 "test4", "test5", "test6", "test7", "test8", "test9",
                 "test10","test11","test12" )]


  p = colSums(sursa_merged %>% select(Metoda2_1:Metoda2_12), na.rm = TRUE)


  procent <- vector("numeric", 12L)
  for (i in 1:n){
    col = paste0("t_r", i, "_r")
    procent[i] <-  paste0( round(p[[i]]*100/sum(sursa_an_curent[,col]),2), "%
")
  }

  luna = c("ianuarie", "februarie", "martie", "aprilie", "mai", "iunie", "iul
ie",
            "august", "septembrie", "octombrie", "noiembrie", "decembrie")
  procent = data.frame(luna, procent)
  return(procent)


  ## write.csv(sursa_merged, "path/sursa_metoda2.csv")

  }
```

Example: Applying the function calculeazaMetodaRaportPerioada() pentru sursa din anul 2016

```
calculeazaMetodaRaportPerioada("sursa2016rev.csv", "sursa2015rev.csv",
                               "q1_m_q3.csv","q1_m_q315.csv")
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4

##           luna procent
## 1    ianuarie   3.7%
## 2   februarie  1.35%
## 3      martie  0.48%
## 4     aprilie  0.92%
## 5         mai  0.84%
## 6       iunie  0.47%
## 7       iulie  0.99%
## 8      august  0.99%
## 9  septembrie   0.5%
## 10  octombrie  1.01%
## 11  noiembrie   0.9%
## 12  decembrie  0.54%
```

The method Comparison with reporting history for the business

Loading libraries

```
library(dplyr)

calculeazaMetodaComparatie = function(sursa_an_curent, sursa_an_precedent){

  sursa_an_curent = read.csv(sursa_an_curent)
  sursa_an_precedent = read.csv(sursa_an_precedent)


  n = 12
  c = 10000

  colnames(sursa_an_precedent)[which(names(sursa_an_precedent) == "ca_12_r")]
= "ca_12_rr"



  sursa_an_precedent[,"media"] = 10 * rowMeans(sursa_an_precedent[, c("ca_01_
r","ca_02_r",
                          "ca_03_r","ca_04_r","ca_05_r", "ca_06_r","ca_07_r",
                          "ca_08_r", "ca_09_r", "ca_10_r","ca_11_r", "ca_12_r
r")])

  sursa_an_curent = left_join(sursa_an_curent, sursa_an_precedent[,c("cod","m
edia")],
                              by = c("cod"))

  for(i in 1:n){
    col1 = paste0("Metoda3_",i)
    col2 = paste0("t_r",i,"_r")
    col3 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")

    sursa_an_curent[,col1] = ifelse(sursa_an_curent[,col2] == 1 &
                        sursa_an_curent[,col3] > sursa_an_curent[,"media"] &
                        sursa_an_curent[,col3] > c , 1, 0)
  }

  p = colSums(sursa_an_curent %>% select(Metoda3_1:Metoda3_12), na.rm = TRUE)
  procent = vector("numeric", 12L)
  for (i in 1:n){
    col = paste0("t_r", i, "_r")
    procent[i] = paste0( round(p[[i]]*100/sum(sursa_an_curent[,col]),2), " %
")
  }

  luna = c("ianuarie", "februarie", "martie", "aprilie", "mai", "iunie", "iul
ie",
               "august", "septembrie", "octombrie", "noiembrie", "decembrie")
  procent = data.frame(luna, procent)
  return(procent)


  ##write.csv(sursa_an_curent, "path/sursa_metoda3.csv")
}
```

Example: Applying the function calculeazaMetodaComapartie() for the VAT dataset, the year 2016.

```
calculeazaMetodaComparatie("sursa2016rev.csv", "sursa2015rev.csv")

##           luna procent
## 1     ianuarie 0.49 %
## 2    februarie 0.78 %
## 3       martie  3.1 %
## 4     aprilie 1.21 %
## 5          mai 1.36 %
## 6        iunie 4.54 %
## 7        iulie  1.8 %
## 8       august 1.85 %
## 9   septembrie 5.82 %
## 10   octombrie 2.19 %
## 11   noiembrie 2.36 %
## 12   decembrie 7.24 %
```

The Method Hidiroglou - Berthelot

Loading libraries

```
library(tidyverse)
library(dplyr)
library(reshape2)
library(lazyeval)
library(stringi)
library(stringr)

calculeazaMetodaHidiroglou = function(sursa_an_curent, sursa_an_precedent){

  sursa_an_curent = read.csv(sursa_an_curent)
  sursa_an_precedent = read.csv(sursa_an_precedent)

  n = 12
  v = 1
  c = 250
  a = 0.05


  colnames(sursa_an_precedent)[which(names(sursa_an_precedent) == "ca_12_r")]
= "ca_12_rr"


  sursa_merged = left_join(sursa_an_curent, sursa_an_precedent[,c("cod","ca_1
2_rr")],
                           by = c("cod"))

  for(i in 2:n){
    sursa_merged$r1 = ifelse(sursa_merged$t_r1_r == 1 & sursa_merged$ca_12_rr
!= 0,
                             sursa_merged$ca_01_r/sursa_merged$ca_12_rr, NA)
    col1 = paste0("r",i)
    col2 = paste0("t_r",i, "_r")
    col3 = paste0(ifelse(i-1 <= 9, "ca_0", "ca_"), i-1, "_r")
    col4 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")

    sursa_merged[,col1] = ifelse(sursa_merged[,col2] == 1 &
                                   sursa_merged[,col3] != 0,
                                   sursa_merged[,col4]/sursa_merged[,col3], NA
)
  }


  mediana = vector("numeric", 12L)
  for (i in 1:n){
    col = paste0("r",i)
    mediana[i] = median(sursa_merged[,col], na.rm = TRUE)
  }


  for(i in 1:n){
    col1 = paste0("t",i)
    col2 = paste0("r",i)
    sursa_merged[,col1] = ifelse(sursa_merged[,col2] < mediana[i],
                                 (sursa_merged[,col2] - mediana[i])/sursa_mer
ged[,col2],
                                 (sursa_merged[,col2]-mediana[i])/mediana[i])
  }


  for(i in 1:n){
    col = paste0('t',i)
    sursa_merged[,col][is.nan(sursa_merged[,col])] = NA
    sursa_merged[,col][is.infinite(sursa_merged[,col])] = NA
  }

  for(i in 2:n){
  sursa_merged$e1 = ifelse(sursa_merged$t_r1_r == 1, sursa_merged$t1 *
                    pmax(sursa_merged$ca_01_r,sursa_merged$ca_12_rr) * v, NA)
    col1 = paste0("e",i)
    col2 = paste0("t_r",i, "_r")
    col3 = paste0(ifelse(i-1 <= 9, "ca_0", "ca_"), i-1, "_r")
    col4 = paste0(ifelse(i <= 9, "ca_0", "ca_"), i, "_r")
    col5 = paste0("t",i)
  sursa_merged[,col1] = ifelse(sursa_merged[,col2] == 1,
                                sursa_merged[,col5] * pmax(sursa_merged[,col4
],
                                                           sursa_merged[,col3
]), NA)
  }

  memory.limit(size=5000)

  metoda5 = function(data){
    for(i in 1:n){
      col = paste0("e",i)
      new_col_name = paste0('Metoda5_', i)

      q1 = quantile(data[,col], probs = 0.25, na.rm = TRUE)
      q2 = median(data[,col], na.rm = TRUE)
      q3 = quantile(data[,col], probs = 0.75, na.rm = TRUE)
      a  = 0.05
      c  = 250

      d1 = max((q2-q1), abs(a*q2))
      d2 = max((q3-q2), abs(a*q2))

      data[,new_col_name] = ifelse((data[,col] < q2 - c*d1 | data[,col] > q2
+ c*d2)
                                   & !is.na(data[,col]), 1, 0)
    }
     return(data)
  }

  # Applying the function metoda5
  sursa_merged = metoda5(sursa_merged)


  sursa_merged = sursa_merged[, !names(sursa_merged)%in%
                c("r1", "r2", "r3", "r4", "r5", "r6", "r7", "r8", "r9", "r10"
, "r11",
                  "r12", "t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8",
                  "t9", "t10", "t11", "t12", "e1", "e2", "e3", "e4", "e5",
                  "e6",  "e7", "e8", "e9", "e10", "e11", "e12")]


  p = colSums(sursa_merged %>% select(Metoda5_1:Metoda5_12), na.rm = TRUE)


  procent = vector("numeric", 12L)
  for (i in 1:n){
    col = paste0("t_r", i, "_r")
    procent[i] = paste0( round(p[[i]]*100/sum(sursa_an_curent[,col]),2), " %
")
  }
  luna = c("ianuarie", "februarie", "martie", "aprilie", "mai", "iunie", "iul
ie",
                 "august", "septembrie", "octombrie", "noiembrie", "decembrie
")
  procent = data.frame(luna, procent)

  return(procent)


  ## write.csv(sursa_merged_lj_mutate, path/sursa_metoda5.csv")
}
```

Example: Applying the function calculeazaMetodaHidiroglou() for the VAT dataset, the year 2016.

```
calculeazaMetodaHidiroglou("sursa2016rev.csv", "sursa2015rev.csv")

##            luna procent
## 1      ianuarie 1.67 %
## 2    februarie 1.44 %
## 3        martie 0.68 %
## 4       aprilie 1.38 %
## 5           mai  1.3 %
## 6         iunie  0.8 %
## 7         iulie  1.6 %
## 8        august 1.58 %
## 9   septembrie 0.77 %
## 10   octombrie 1.62 %
## 11   noiembrie 1.49 %
## 12   decembrie  0.8 %
```

Benford's Law

Loading libraries

```r
library(ggplot2)
```

The Graph illustration for the Benford's Law

```r
compareBenford <- function(coloana){
  digits <- coloana[!is.na(coloana)]
  digits <- substr(stringr::str_extract(as.character(abs(digits)),
                                        pattern = "[^0\\.]"),1,1)
  digits <- factor(digits, levels = 1:9) # ensure all digits represented
  depth <- prop.table(table(digits))
  ben <- log10(1 + (1/(1:9)))
  dat2 <- data.frame(ben, depth)
  names(dat2) <- c("Benford","Digit",deparse(substitute(coloana)))
  dat2L <- reshape2::melt(dat2,id.vars="Digit", variable.name = "Type",
                          value.name = "Frequency")
  ggplot(dat2L, aes(x=Digit, y=Frequency, fill=Type)) +
  geom_bar(stat = "identity", position = "dodge")
}
```