# Optimising the calculation of statistical functions

André Rodrigues*, Carla Silva,
Paulo Borges and Sérgio Silva

NLPC Lda.,
Praça Mouzinho de Albuquerque, 113 – 5º,
4100-359 Porto, Portugal
Email: razor336@gmail.com
Email: carla.maps@gmail.com
Email: paulo.borges@nlpc-incta.com
Email: sjoaosilva@yahoo.com
*Corresponding author

Inês Dutra

Department of Computer Science,
CRACS INESC TEC and University of Porto,
Rua do Campo Alegre, 1021,
4169-007, Porto, Portugal
Email: ines@dcc.fc.up.pt

**Abstract:** Statistical data analysis methods are well-known for their difficulty in handling large number of instances or large number of parameters. In this paper, we study popular and well-known statistical functions, generally applied to data analysis, and assess their performance as implemented by SPSS, MATLAB, R and our own software, DataIP. We use medium to large datasets and show that DataIP outperforms SPSS, MATLAB and R by several orders of magnitude. We argue that the design and implementation of these functions need to be rethought to adapt to today's data challenges.

**Keywords:** data analysis; statistical functions; performance evaluation; SPSS; MATLAB; R.

**Reference** to this paper should be made as follows: Rodrigues, A., Silva, C., Borges, P., Silva, S. and Dutra, I. (xxxx) 'Optimising the calculation of statistical functions', *Int. J. Big Data Intelligence*, Vol. X, No. Y, pp.xxx–xxx.

**Biographical notes:** André Rodrigues obtained his MSc in Computer Science from the University of Porto, Portugal in 2015. He is pursuing his PhD in Computer Science, also at the University of Porto. His main research interests are parallel algorithms and techniques, and code restructuring and optimisation. Currently, he has been developing software for the NLPC Dataias company and is responsible for the most recent version of the software used in this paper.

Carla Silva obtained a Scientific-Technological Diploma in Informatics at the Colegio Internato dos Carvalhos, Portugal, then proceeded to Engineering Sciences – Surveying Engineering at Faculty of Sciences, Porto University, Portugal. Later, she did a research project in ICT in Business at the Leiden Institute of Advanced Computer Science, The Netherlands. Recently she worked at the Center for Research in Advanced Computing Systems, INESC TEC, Portugal. She worked for the NLPC Dataias company and developed most of the initial code for the statistical functions evaluated in this paper. Her main interests are computational agent systems, decision support systems, positioning and satellite navigation, data mining and machine learning. She has one book chapter at Springer and one paper at IEEE.

Paulo Borges obtained his MSc in Physics Engineering from the University of Coimbra, Portugal in 1997. He worked as a researcher at the Optical Technologies Center at the University of Porto in Optical Fibers from 1997 to 1999. From 2000 to 2011, he worked as a Software Consultant, and participated in several projects in various sectors of activity and in various companies and organisations such as: Barclays Bank, TCS, DELPHI Corporation, CSC, SIBS, Millenium BCP, BPI, BFA, Sodecia, Efacec, Sage, Sofin, EccoleâŁ™t, First Solutions, among others. In 2011, he was one of the founders of NLPC Institute in Behavioral Science and Applied Psychology, leading the area of research and development in the area of decision support systems. He participated in various training programs in information systems and project management. He has extensive experience in systems architecture, project management and software engineering.

Sérgio Silva obtained his degree and Master's in Mathematics from the University of Minho, Portugal, and currently collaborates on research and development projects, under his PhD in Intelligent Software Systems at the University of Vigo, Spain. He has worked as a Professor of higher and further education and participated in research projects related to mathematics, statistics and information technology. His main large areas of research are data analysis, data mining, text mining and big data. He has published several articles and has participated in conferences in the field of statistics.

Inês Dutra is a Professor of the Department of Computer Science, School of Sciences of the University of Porto. She obtained her PhD degree from the University of Bristol, England in 1995. She worked at the Universidade Federal do Rio de Janeiro, from 1985 to 2007. In 2007, she moved to Portugal. Her main research interests are logic programming, inductive logic programming, data mining, applications in medical informatics and parallelisation. She has published more than 50 papers in main conferences and journals, and supervised several MSc and PhD students. She is a researcher consultant to NLPC Dataias.

This paper is a revised and expanded version of a paper entitled 'Performance evaluation of statistical functions' presented at DataCom 2015 Conference, Chengdu, China, 19–21 December 2015.

## 1 Introduction

As the amount of information grows more than exponentially along the years, we start to face several challenges. In order to process and analyse all these data, we need new tools and algorithms that can cope with their heterogeneity and volume. Statistical data analysis methods are well-known for their difficulty in handling large number of instances or large number of parameters. Often just accessing the data may be a serious bottleneck. These considerations motivated work on the so-called 'noSQL' data-bases, such as Google's BigTable (Chang et al., 2008), or the Apache Foundation's Spark (Zaharia et al., 2010), which support distributed data. Other works in the direction of improving data partitioning and memory utilisation, not directly related with the implementation of statistical functions, but dedicated to big data handling are Slagter et al. (2013, 2015) and Hsu et al. (2015). One remarkable work and somewhat related to ours is Kane et al.'s (2013) implementation of the R packages *foreach* and *bigmemory*, that allow the implementation of statistical functions in parallel. Following this trend, MATLAB has also been investing on new tools for efficient data analysis and processing (http://www.mathworks.com/solutions/data-analytics/, accessed in September 2016).

In contrast to transaction oriented data-bases, statistical methods are often applied to data repositories with read-only data, which avoids data consistency overheads. Progress in DRAM technology has enabled manipulating large amounts of data in main memory. However, these solutions require specialised hardware and software to be able to handle massive amounts of data. In general, in order to analyse data, most people use traditional tools such as Excel, SPSS, MATLAB or R. Although these are very popular, they are not tailored to handle large amounts of data. In this work, we show that we can process and analyse millions of instances, with potentially hundreds of variables, faster than these systems, with the same simplicity, and without the need of specialised resources, except the use of top-of-the-shelf multicore machines.

We concentrate on the following bivariate and multivariate analysis, which are most popular in data analysis: Chi-square and Shapiro-Wilk tests, Pearson, Spearman and Kendall correlations, linear, logarithmic, 2nd and 3rd order polynomial regressions, Wilcoxon, Kolmogorov-Smirnov (K-S), T-student and Bartlett statistical tests, Pearson, Spearman and Kendall significance tests, ANOVA and principal component analysis (PCA). We also study the performance of calculating data summaries and loading data. With the exception of loading data, our results are always superior to SPSS, MATLAB and R, and our best result, for Kendall, is several orders of magnitude better than SPSS, MATLAB or R.

Our software, DataIP, is written in C/C++, and implements the aforementioned functions. In the next sections, we describe the main functions our software implements, discuss about implementation issues, and present our experimental methodology and results.

## 2 Background

One important step in the knowledge discovery process is the statistical analysis of data. Usually, we start to 'know' the data using descriptive and inferential statistics through univariate, bivariate and multivariate methods. Later, we may go through other important steps related with machine learning techniques to build more powerful data models, but in this work we focus solely on the statistical analysis.

The major purpose of univariate analysis is to describe the data. Univariate analysis is usually used in the first descriptive stages of problem solving, being complemented by more advanced, inferential bivariate or multivariate analysis. Descriptive statistics (Rumsey, 2010) describe and summarise data. Univariate descriptive statistics describe individual variables. Exploratory analysis of data gives us a

summary statistics of measures of central tendency, dispersion and shape of the data.

Measures of central tendency locate a distribution of data along an appropriate scale, such as: geometric mean, harmonic mean, arithmetic average, median and mode. The purpose of measures of dispersion is to find out how spreads out the data values are. Another term for these statistics is measures of spread: interquartile range, percentiles, average absolute deviation (or simply called average deviation), range, standard deviation and coefficient of variation. This analysis is usually followed by measures of shape. The measures of shape indicate the symmetry and flatness of the distribution of a data sample. A distribution of data item values may be symmetrical or asymmetrical. In this field, we have: variance, kurtosis, central moment of 3rd order and Pearson asymmetry coefficients G1 and G2.

Bivariate analyses are conducted to determine whether a statistical association exists between two variables, the degree of association if one does exist, and whether one variable may be predicted from another. It deals with causes or relationships. The major purpose of bivariate analysis is to:

1 define the nature of the relationship

2 identify the type and direction of the relationship

3 determine if the relationship is statistically significant

4 identify the strength of the relationship.

In bivariate analysis, we use non-parametric statistics (Conover, 2006), given that our focus is on inferential statistics (Casella and Berger, 2008). We use the same hypotheses testing used in the univariate analysis, as well as other tests specific to bivariate analysis (K-S, Wilcoxon and T-test).

Multivariate studies are analogous to bivariate studies, but involve multiple variables. Researchers could then use multivariate statistical analysis to study the relationships between all of the variables. Multivariate analytical techniques represent a variety of mathematical models used to measure and quantify outcomes, taking into account relevant factors that can cause this relationship. The most common is multiple regression analysis (Montgomery and Runger, 2010) whose objective is to understand how the value of the dependent variable changes when any of the independent variables is varied, while the other independent variables are fixed, using multiple response variables. With regression (Wonnacott and Wonnacott, 1990), we attempt to find a function which models the data with the minimum error.

In PCA (Venables and Ripley, 2002), our intent is to reduce data dimensionality. In order to perform that task, we centre the data and calculate the covariance matrix and extract the eigenvectors, using singular value decomposition on the covariance matrix. The calculated principal components are orthogonal and less or equal the number of input variables.

## 3 Implementation

Based on the whole data analysis process, we developed a framework that can make use of tools for univariate, bivariate, multivariate statistical analysis and also include some machine learning methods. In this paper, we concentrate on the statistical methods only. In a general context, knowledge extraction is performed through exploratory analysis, univariate, bivariate and multivariate, and using descriptive and inferential statistics. We have optimised sequential and parallel implementations of the methods. Moreover, we take advantage of the fact that several statistical functions share common data preprocessing operations, and execute them only once to speedup execution. For example, correlation functions usually sort the two variable values to be correlated. We sort all variables beforehand, store the resulting vectors, and reuse them whenever needed. We also store minimum and maximum values, modes and medians, among others. Added to these optimisations, we also preprocess all data variables by categorising them as much as possible as numerical or categorical, continuous or discrete.

The algorithms were developed in C/C++ and implement well-known functions used in SPSS, R and MATLAB. All functions were implemented to produce the same results (including formatting) as R, except kurtosis, which was implemented according to MATLAB because R needed an extra library to run it.

The following is handled by our implementation.

### *3.1 Univariate analysis*

- Dataset dimensions and missing values are determined.
- Variable types are automatically detected. Some statistical functions are only applied to the numerical data.
- Measures of dispersion are calculated which results in the output of a shape which includes the mean, median, mode, maximum and minimum and some quartiles (the first and the third).
- Variance, standard deviation and mean absolute deviation (average deviation from the average) are calculated. We calculate the arithmetic, geometric and harmonic means.
- The mode is the most common value obtained in all the observations. If they are all different, we do not present the mode.
- Interquartile range and the difference between the upper and lower quartiles are calculated. The total amplitude or range, the difference between the maximum and minimum values, is also calculated.
- Calculation of asymmetry measures, with the results: positive asymmetrical when the median is less than the average and higher than the mode, asymmetric negative in cases where the median is less than the mode and

higher than the average and symmetrical when the median is equal to mode and the average.

- The following are also calculated: kurtosis, 3rd moment (calculation as shape measures), G1, G2 and the coefficient of variation.
- Outliers are identified consisting of elements that are above the 3rd quartile + 1.5 × interquartile range and below the 1st quartile – 1.5 × interquartile range (the 1.5 is used because John Tukey, the inventor of the box-and-whisker plot in 1977, picked 1.5 × IQR and this has worked well since then. For our purposes, the choice of this value would not affect the results).
- To end the univariate analysis, we calculate the Chi-square and the Shapiro-Wilk test, to check whether the sample has a normal distribution.

### *3.2 Bivariate analysis*

For the bivariate analysis, we calculate correlation coefficients (Kendall and Gibbons, 1990) indicating the strength and direction of a linear relationship between two random variables, using three methods: Pearson, Spearman and Kendall. Pearson gives the direct relationship between a pair of variables while Spearman and Kendall give correlation coefficients for the rank function between two variables. For Pearson, the values obtained through the measures of association always vary between –1 and 1, where 1 indicates that there is a perfect positive correlation between two variables, –1 a negative perfect correlation, and 0 both variables are not linearly dependent on one another. The pairs of variables whose correlations are greater than or equal to 0.75 are considered strong correlations. For these pairs, we produce a bivariate or simple linear regression, using the least squares method. For Spearman and Kendall, the variable values are not important. The Spearman correlation may still give values of –1 when Pearson will give greater values, if the variables are inversely proportional, but they do not decrease by the same amount. Conversely, Spearman may give values of +1 when Pearson will give smaller values, if the variables grow proportionally, but not by the same amount. The differences among the outcomes of these three tests are relevant when performing data analysis, but this discussion is out of the scope of this work [for more details about Spearman and Kendall rank correlations, a good reference is Croux and Dehon (2010)].

The bivariate analysis seeks to obtain a mathematical model that best fits the observed values of Y due to the variation of the variable levels of X. We then have a simple linear data model. Once the coefficients of this linear model are found and the respective errors are standardised, tests are applied to validate the model. Usually, the main test applied is the t-test, but depending on the data distribution, the t-test may not be suitable and another test needs to be applied. As output, we have the standard error of the residuals and the respective degrees of freedom, multiple R-squared and adjusted R-squared. The F-statistic is also calculated. Residuals are presented as follows: minimum, 1st quartile, median, 3rd quartile and maximum. A normality test, Shapiro-Wilk, is applied on the residuals, where through the p-value we validate the model as good or bad. A summary of the errors is then performed to study how well those variables are correlated, and how good the prediction is given by the regression model. In this phase, we repeat the same process we used with the univariate analysis, but now on the regression errors. We calculate: the average error, the mean square error, the root mean square error and the normalised root mean square error and the mean absolute error. We perform non-parametric statistical tests (Hollander et al., 2013) such as the Wilcoxon rank sum test, the two-sample K-S test (Marsaglia et al., 2003) and the Welch two-sample t-test. We also compute F ANOVA in order to study the variance. Nonlinear models are also generated: exponential, logarithmic, quadratic and cubic wherein the validation of models follows the same procedure as the linear models.

### *3.3 Multivariate analysis*

Next, multiple linear regression is implemented, which involves more than two variables, giving rise to the multivariate study. The employed multivariate statistic allows the prediction values of one or more response variables (dependent) from several independent or predictive variables. The correlations between multivariate variables are calculated using regression, as well as residuals and the regression equation and its coefficients. A multivariate analysis of variance and error, similar to the one applied in simple linear regression, is performed. A regression is calculated for each variable. Then the Bartlett test is performed on the homogeneity of the variances, as well as the Kruskal-Wallis rank sum test. The multivariate analysis ends with a factor analysis where we calculate the principal main components using PCA, with a preliminary study of the Kaiser-Meyer-Olkin (KMO) index, which must be greater than 0.5 in order that we can proceed to the PCA. PCA is performed using the singular value decomposition technique (Press, 2007).

Input data to our framework can be in the form of comma-separated values (CSV) files or can be from a relational database. If the data comes from a database connection, a query builder allows the user to apply filters and perform queries, which will, by their turn, generate CSV files. We parse the final CSV file, and transform it in a C structure that will be analysed using the steps described. Frequencies and modes for qualitative attributes are calculated. We also perform a cluster analysis (non-hierarchical) using k-means, and a hierarchical agglomerative cluster analysis using similarity measures such as the single linkage method, complete linkage, average and centroid linkage. Cluster analysis is not included in this work.

We implemented all of those methods from scratch, in C/C++, using the best algorithms found in the literature, optimising for performance, and removing redundant calculations. For example, as we have all functions

integrated, lots of calculations can be saved because they are common to several methods. One example is sorting variable values, which is performed just once. Memory usage is other important issue, and we needed to organise our implementation to minimise writes and memory allocation during calculations (reads are always faster than writes). Another issue on the implementation is the complexity of the algorithms. Our implementations keep a maximum complexity of $\mathcal{O}(n \log n)$.

In order to get the most possible organised structure and layer separation, we are using an object oriented structure in C++.

Our most important optimisation is on the calculation of Kendall $\tau$ (correlations). This is known to have quadratic complexity, but we use Knight's algorithm (Knight, 1966; Christensen, 2005), that calculates Kendall $\tau$ with $\mathcal{O}(n \log n)$ complexity. An implementation of Knight's algorithm is also found in the pcaPP R package through the function `cor.fk`.

In order to further optimise our implementation and allow it to take advantage of multicore machines, we parallelised the code using openMP (Chandra, 2001). In order to get big chunks of processing and always get speedups avoiding overheads, parallelism is used on the following steps:

- On data loading when checking for missing values, sample dimension and types of variables.
- On calculating the data summary, all statistics related to this procedure are calculated in parallel, attribute by attribute. If we have only one attribute, the calculations are sequential, but if we have more than one, they are performed in parallel.
- On calculating correlations, all pairs are calculated in parallel. Each core processor calculates one pair with all three correlations (Pearson, Spearman and Kendall).
- On multivariate analysis the idea is the same. Each core calculates all analysis to each one of the permutations.

## 4 Experimental methodology

We performed our experiments in three different machines: a plain multicore workstation with 8 cores AMD FX 350, 16 GBytes of memory and Linux Fedora 20 (M*achine 1*); a Xeon server with 8 cores Intel Xeon x5550 (16 threads), 24 GBytes of memory and Fedora 20 (*Machine 2*); an IBM system x3755 with 6 cores, 48 GBytes of memory; and a Windows Server 2012 Datacenter (*Machine 3*).

In order to study the scalability of the systems, we used two datasets: real data collected from Hospital das Clínicas (São Paulo, Brazil), consisting of around 200,000 patient discharges (*Dataset 1*) with 26 variables, from which we synthetically created other datasets of increasing sizes, and a synthetic dataset created with millions of instances, but with only two variables because of memory constraints (this dataset alone is almost 5 GBytes large) (*Dataset 2*).

We use Dataset 1, patient discharges, with different versions:

- 300 version: It is a subset with 300 rows, and nine numeric variables without nulls.
- 200 k version: original dataset consisting of 201,879 discharge patients, and six numeric variables without nulls
- 1 M version: it is a replication in chunks of the original dataset 200 k in order to achieve one million instances, with six numeric variables without nulls.
- 5 M version: it is a replication in chunks of the original dataset 200 k, in order to achieve five million instances, with six numeric variables without nulls.
- 10 M version: it is a replication in chunks of the original dataset 200 k, in order to achieve ten million rows, with six numeric variables without nulls.

Dataset 2 is a synthetic dataset created to also study the scalability of our implementation when handling many millions of instances. It has only two variables, due to the memory limits of our machines. The dataset was generated with random numbers between 0 and 1,235. This range was arbitrarily chosen. The choice of random numbers does not affect the results. They could be also real numbers. The file size is 4.8 GBytes.

In order to check dataIP versatility and how it handles larger number of variables, we also used two datasets from the University of California at Irvine Machine Learning Repository, quite popular in the literature. One of them is CoverType (*Dataset 3*), consisting of 55 variables and 581,012 instances. The other one is YearPredictionMSD (*Dataset 4*), consisting of 91 variables and 515,345 instances.

Input data to our framework can be in the form of CSV files or can be retrieved from a relational database. In this work, we used CSV files for DataIP, MATLAB and R and SAV format for SPSS. In DataIP, the CSV file is parsed, frequencies and modes for qualitative attributes are calculated, and attribute types are detected.

Most of our experiments were run with Dataset 1 using Machine 1. Datasets 3 and 4 were also run using Machine 1. Dataset 2 ran in the last two machines (the dual Xeon and the IBM servers, Machine 2 and Machine 3, respectively). The experiment in Machine 2 compares the Kendall results running on Linux and Windows with only one thread and a larger set of instances, as in Dataset 2. The experiment in Machine 3 compares the execution of Kendall on Oracle with our implementation. Kendall is highlighted here because we use an algorithm whose complexity is $\mathcal{O}(n \log n)$ as compared with the quadratic implementations of MATLAB or Oracle.

We performed experiments for the summary, Chi-square and Shapiro-Wilk tests, the three correlations (Pearson, Spearman and Kendall), three significance tests (Pearson, Spearman and Kendall), linear and nonlinear second and third order polynomial and logarithmic regressions,

Wilcoxon, K-S, T and Bartlett statistical tests, and PCA. Some experiments were aborted by SPSS, MATLAB or R, because of lack of memory or because they exceeded our maximum running time limit. Experiments with SPSS, MATLAB and R were run with default parameters. Some of the functions implemented by this software are calculated in different forms and produce different results. Examples are: Shapiro-Wilk in SPSS also calculates K-S and shows descriptive analysis; also SPSS K-S and Wilcoxon produce graphical output files (jpg), which may cause them to be slower than the other software; Bartlett outputs other information such as covariance and correlations, among others.

All experiments in Machine 1 and Dataset 1, Dataset 3 and Dataset 4 were run with 1 and 8 threads. In our implementation, we measured execution times necessary to execute each one of the statistical functions, but because some operations need to be computed only once (for example, sorting variable values), this time is computed only once. A list of functions timed in MATLAB is: `textscan`, `corr`, `anova2`, `kmeans`, `fitlm`, `min`, `max`, `median`, `mode`, `mean`, `geomean`, `harmmean`, `std`, `var`, `range`, `iqr`, `mad`, `quantile`, `prctile`, `kurtosis`, `moment`, `ranksum`, `kstest2`, `chi2gof` and `ttest2`. A list of functions timed in R is: `read.csv`, `chisq.test`, `shapiro.test`, `cor`, `cor.test` (Hollander and Wolfe, 1973), `aov` (Chambers et al., 1992), `prcomp`, `princomp`, `lm`, `summary`, `min`, `quantile`, `median`, `mean`, `max`, `var`, `sd`, `IQR`, `range`, `kurtosis`, `wilcox.test`, `ks.test`, `t.test`, `bartlett.test`, `moment`, and `plot`. In SPSS, we used `GET FILE`, `CORRELATIONS`, `RANK`, `NONPAR CORR PRINT = KENDALL`, `T-TEST PAIRS`, `REGRESSION`, `NPTESTS INDEPENDENT TEST MANN_WHITNEY KOLMOGOROV_SMIRNOV`, `NPAR TESTS CHISQUARE`, `GLM PRINT = TEST(SSCP) RSSCP`, `EXAMINE`, and `FACTOR`.

Shapiro-Wilk was not run for all dataset sizes, because it is recommended to be applied to input sizes of at most 5,000 rows not to lose precision (Royston, 1995). Linear and logarithmic regression, 2nd and 3rd order polynomial regression, the Wilcoxon, K-S, T-student and Bartlett statistical tests, Pearson, Spearman and Kendall significance tests and ANOVA are only applied to datasets that have strong correlations between variables (correlation value greater than 0.75 with confidence greater than 95%). This is true only for the 300 dataset.

For Machine 1, in total, we performed 206 experiments: 35 for SPSS, 31 for MATLAB, 44 for R, and 48 for DataIP (1 and 8 threads). Most of them were repeated 30 times and averages of execution time were taken. The only exception was Kendall, which was repeated only ten times due to its very long execution time in SPSS, R and MATLAB. For MATLAB, in particular, we ran 31 times and discarded the first run, since it was slower than the remaining ones. Except for that, variation among execution times for each run was negligible.

Our version of SPSS (academic license, authorised user) does not provide any function to assess execution time. We used the python interface to perform this task. This proved to be a very tedious task, since some of the code could not run through 'Run Script', and our license did not provide the means to perform the experiments in batch. Moreover, SPSS does not allow Spearman correlations to run all variables at once. We had two alternatives: to use CROSSTABS or RANK variables. CROSSTABS limits the number of instances to 1,000. We were forced to use RANK. Except for SPSS, all runs were performed in batch.

Data input was not taken into account when timing each statistical function. We measured data input times separately.

Except for R, whose code is written in C, we would expect SPSS and MATLAB, that use Java coding, to be slower than DataIP, which is also written in C/C++ like R.

All execution times include time spent to write files in disk. All times are reported in seconds and because of the differences in magnitude, are presented with nine decimals.

## 5 Experiments and results

### *5.1 Dataset 1*

Tables 1 to 11 show execution times, in seconds, for each statistical function, using our implementation (DataIP (1) – with a single thread, and DataIP (8) – with 8 threads), SPSS, MatLab and R, running on Machine 1, applied to datasets 300, 200 k, 1 M, 5 M and 10 M.

The minimum average execution time was obtained by DataIP (8 threads), running the Bartlett test for our 300 dataset (0.000001709 seconds) and the maximum was obtained by MATLAB (4931.319953 seconds), running the Kendall correlation for the 200 k dataset (it could not run the larger datasets).

#### *5.1.1 Data input*

Table 1 shows data input times for all software.

**Table 1** Load (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.020093200 | 0.022729400 | 0.023347700 |
| MATLAB | 0.003309000 | 2.661653000 | 13.514717000 |
| R | 0.009000000 | 6.304000000 | 14.403000000 |
| DataIP (1) | 0.018793400 | 2.294150000 | 12.600600000 |
| DataIP (8) | 0.019221200 | 1.008460000 | 5.146790000 |
| | *5 M* | | *10 M* |
| SPSS | 0.022880760 | | 0.022780460 |
| MATLAB | 136.083043000 | | - |
| R | 59.249000000 | | 119.781000000 |
| DataIP (1) | 70.003900000 | | 146.975000000 |
| DataIP (8) | 28.128700000 | | 56.732000000 |

SPSS performs a pretty decent job when reading data, maintaining a constant input time no matter the data size. It may be taking advantage of using its own file format, because we provide input files in SAV format. We will see later that, although it performs well when loading files, it has the poorest performance when calculating the statistical functions. MATLAB takes a quite long time to input data as the file sizes increase and does not manage to read multiple times the largest data file with around 10 million instances. This leads us to conclude that MATLAB will be unable to load larger data files. Our MATLAB script runs one session to input data running `textscan` 30 times. If we start MATLAB 30 times (30 different sessions), where each one executes a textscan, the input times drop considerably, being close to SPSS times (0.006222, 0.010844, 0.010858, 0.010711, and 0.010842 seconds for 300, 200 k, 1 M, 5 M, and 10 M, respectively). DataIP(1) is competitive with R up to the 1 M data size. As the data sizes increase to 5 M and 10 M, DataIP becomes a bit slower than R (note that DataIP preprocesses the data while loading). This difference disappears and times drop to almost half of R's when we use 8 threads for the DataIP loading function. Because preprocessing can be performed in parallel, DataIP can be faster than R (even though it performs other actions such as checking variable types and calculating modes and frequencies). In any case, as we will show next, the advantage that R and SPSS have over DataIP on loading, disappears when one needs to perform some calculation with the data. It is important to note that DataIP is not fully optimised and that the results with 8 threads show that the implementation can still scale to larger datasets. If we disable the preprocessing of data types, the DataIP loading times become linear with a small slope, as we increase the data size. However, besides having the advantage of computing only once certain functions (like sorting), preprocessing also has the advantage of preventing execution errors. For example, geometric mean should not be computed if a variable has negative values. While MATLAB requires that the user handles these variables (otherwise the system triggers an error), dataIP skips them when calculating the geometric mean.

Figure 1 shows the behaviour of all software when performing data input.

Figures 2, 3 and 4 show examples of codes that perform data input. The SPSS code is embedded in Python, as explained earlier.

### *5.1.2 Summary*

Table 2 shows execution times for SPSS, MATLAB, R and our implementation of the summary for 1 thread [DataIP (1)] and for 8 threads [DataIP (8)], with varying dataset sizes. All experiments were run on Machine 1. For this task, MATLAB is twice as fast as R. DataIP (1) is almost 50 times faster than R (for the smallest dataset) and almost 20 times faster for the largest (10 M). DataIP (1) is also faster than MATLAB (varying from 6 to 56 times faster, depending on the dataset size).

**Figure 1** Load data execution times (seconds) (see online version for colours)

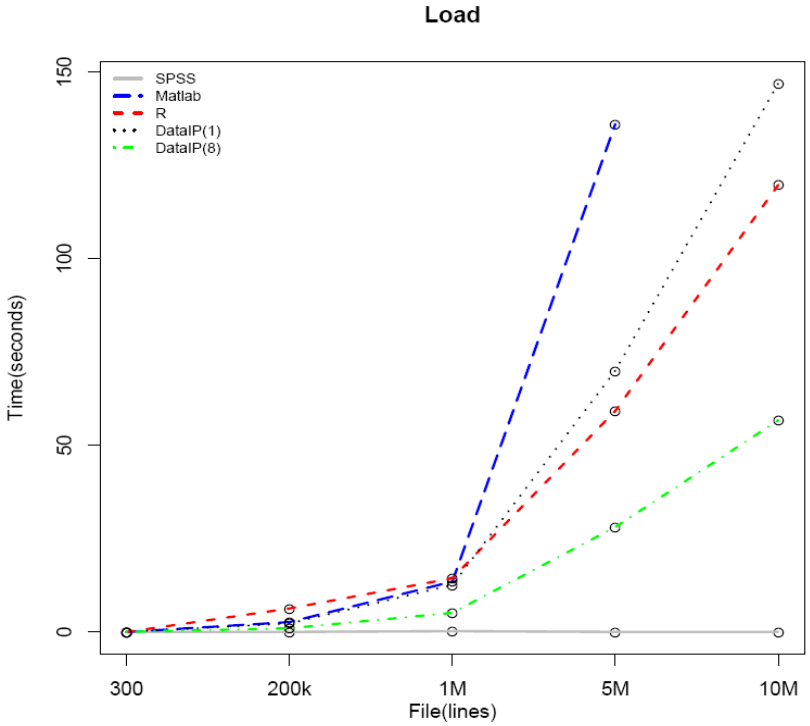


**Figure 2** SPSS example script for data input

```
BEGIN PROGRAM.
import spss, time
total = 0
for i in range(31):
    start = time.time()
    file="".../10M.sav"
    varlist="cod_alta cod_hosp cod_sexo
    cod_ocup cod_catint cod_mnibge"
    spss.Submit(""""
    GET FILE='%s'.
    """ %(file))
    corTime = time.time() - start
    if i != 0:
       total += corTime
    print corTime
print "Average: ", total/(i-1)
spss.Submit("""OUTPUT EXPORT
          /TEXT
          DOCUMENTFILE='output_10M_0.txt'.
""")
END PROGRAM.
```

**Figure 3** MATLAB example script for data input

```
fileID = fopen('M10M.txt','w');
sumtime=0;
for i = 1:31
  tic
  file = fopen('10M.csv');
  out = textscan(file, '%f %f %s %s %f %s
  %f %f %s %s %s %s %s %s \
    %s %s %s %s %f %s %s %s %s %s %s
    %s','delimiter', ';',
    'HeaderLines',1);
  fclose(file);
  cod_alta=out{1};
  cod_hosp=out{2};
  cod_sexo=out{5};
  cod_ocup=out{7};
  cod_catint=out{8};
  cod_mnibge=out{19};
  toc
  if (i~=1) sumtime=sumtime+toc; end
end
fprintf(fileID,'%f\n',sumtime/30);
fclose(fileID);
```

DataIP does not improve the performance much when using 8 threads, achieving a maximum speedup of 2. Note that all DataIP times take into account the output to disk, which can be very time consuming. Our parallelisation is very simple and only computes in parallel the summary of each attribute. As these datasets have a small number of non-null attributes, we do not take full advantage of the multi-threaded version, but still manage to have an efficiency of almost 50% for all larger data sizes, doubling the speed of the sequential version.

**Figure 4** R example script for data input

```
output <- file("R10M.txt", open = "wt")
sink(output)
results <- function(){
  data<-
  read.csv("10M.csv",header=F,sep=";")
  data<-data[2:length(data[,1]),]
  cod_alta<-
  as.numeric(as.vector(data[,1]))
  cod_hosp<-
  as.numeric(as.vector(data[,2]))
  cod_sexo<-
  as.numeric(as.vector(data[,5]))
  cod_ocup<-
  as.numeric(as.vector(data[,7]))
  cod_catint<-
  as.numeric(as.vector(data[,8]))
  cod_mnibge<-
  as.numeric(as.vector(data[,19]))
}
system.time(replicate (30,results()))/30
sink()
```

We do not have access to a parallelised MATLAB toolbox, but, even if the 8-threads version of MATLAB had perfect speedup, DataIP with 8 threads would still be almost 32 times faster than MATLAB with 8 threads to calculate the summary for the 300 dataset, and 115 times faster for the larger 5 M dataset. The same effect in a different scale happens when comparing DataIP with R.

**Table 2** Summary (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.025017508 | 0.848454031 | 4.843381265 |
| MATLAB | 0.011257000 | 0.396472000 | 2.021464000 |
| R | 0.010033330 | 0.866366667 | 4.124233000 |
| DataIP (1) | 0.000199484 | 0.023866800 | 0.319406000 |
| DataIP (8) | 0.000352226 | 0.023766300 | 0.215104000 |
| | *5 M* | | *10 M* |
| SPSS | 29.332320517 | | 64.272009676 |
| MATLAB | 10.006390000 | | - |
| R | 21.997370000 | | 48.257370000 |
| DataIP (1) | 1.416320000 | | 2.784620000 |
| DataIP (8) | 0.743863000 | | 1.401760000 |

**Figure 5** Summary execution times (seconds) (see online version for colours)

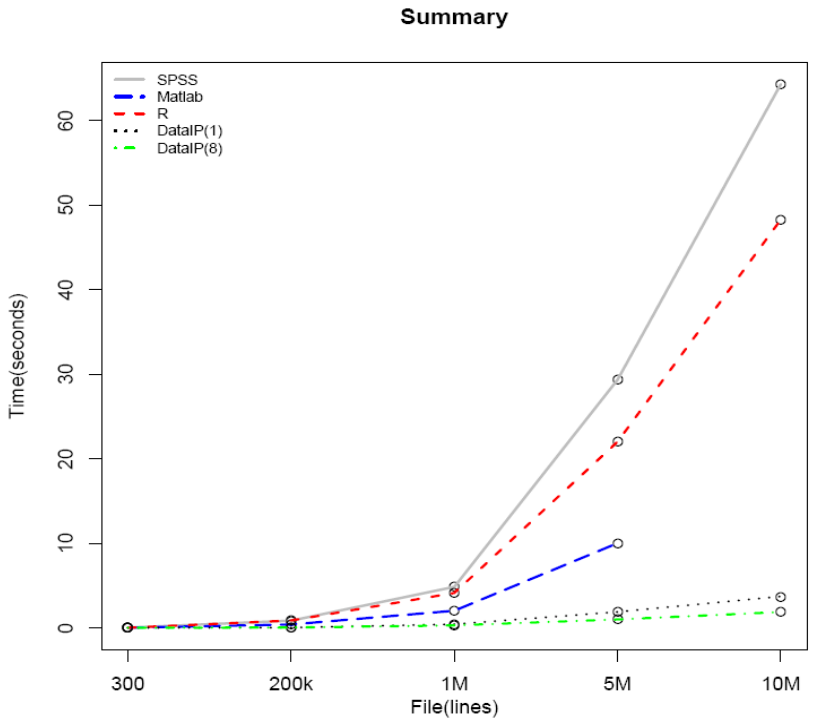



**Figure 6** Chi-square test execution times (seconds) (see online version for colours)

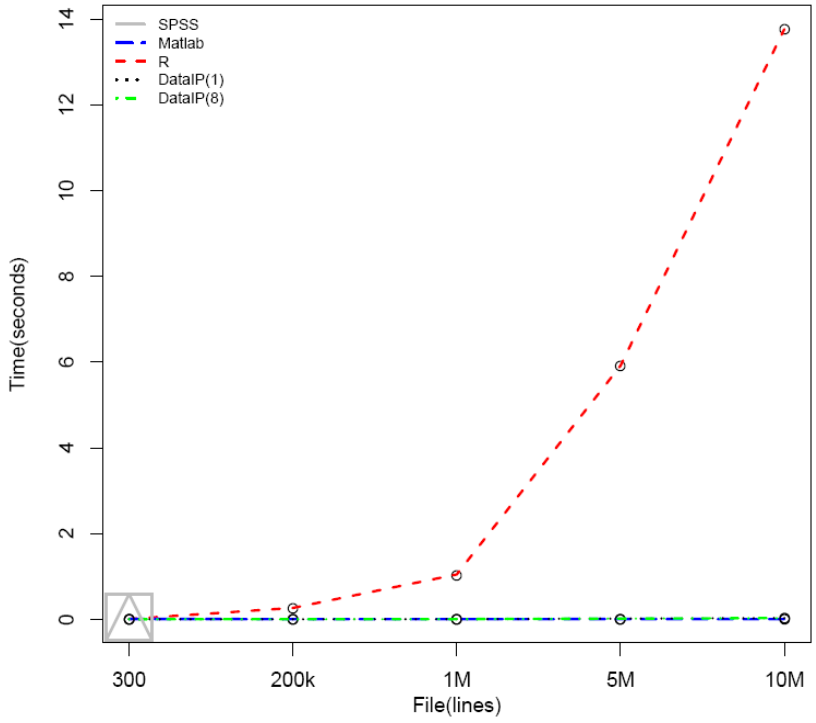



Figure 5 shows how the three implementations compare in terms of execution times, in seconds, as we vary the dataset sizes.

### *5.1.3 Chi-square test*

The chi-square test is timed for all software, even though MATLAB's `chi2gof` produces results that are different from R and from SPSS. As DataIP is implemented to give

results as in R, it produces results like SPSS and R. Average execution times are shown in Table 3.

SPSS could not perform the test for our larger datasets due to a Java out of memory error (heap space). The default heap size available at our configuration file (jvmcfg.ini) is 2 GBytes, already quite high.

**Table 3** Chi-square test (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.045724256 | - | - |
| MATLAB | 0.005219000 | 0.003709000 | 0.003753000 |
| R | 0.006466667 | 0.264633333 | 1.045500000 |
| DataIP (1) | 0.000018806 | 0.000592726 | 0.003258870 |
| DataIP (8) | 0.000018306 | 0.000664484 | 0.003357810 |
| | *5 M* | | *10 M* |
| SPSS | - | | - |
| MATLAB | 0.004391000 | | 0.003863000 |
| R | 5.916433000 | | 13.769167000 |
| DataIP (1) | 0.014693000 | | 0.029212400 |
| DataIP (8) | 0.014883000 | | 0.029014700 |

MATLAB does a good job with the Chi-square test maintaining constant performance even for the larger datasets. R performs very poorly while DataIP performs in between, but with worse performance than MATLAB as we increase the dataset size. Once more this effect is due to the fact that the time taken to write results to disk is being taken into account for DataIP. Also for this reason, DataIP can not achieve speedups even executing the data preprocessing in parallel. For this function, we got better performances than for the summary.

Figure 6 shows how the three implementations compare in terms of execution times, in seconds, as we vary the dataset sizes. R has exponential behaviour, while MATLAB and DataIP have linear behaviour.

### *5.1.4 Shapiro-Wilk test*

The Shapiro-Wilk test performed much better in DataIP than in R (more than 100 times faster). This function is not directly implemented in MATLAB, therefore no results are reported. We only ran this for the 300 dataset as it is recommended only for datasets up to 5,000 instances.

**Table 4** Shapiro-Wilk test (time in seconds)

| | *300* |
|---|---|
| SPSS | 3.277540948 |
| R | 0.005933333 |
| DataIP (1) | 0.000050726 |
| DataIP (8) | 0.000059871 |

### *5.1.5 Pearson correlation*

Figure 7 shows the behaviour of all software when running the Pearson correlation. SPSS, MATLAB and R have exponential behaviour while DataIP has linear behaviour.

**Figure 7** Pearson correlation (see online version for colours)

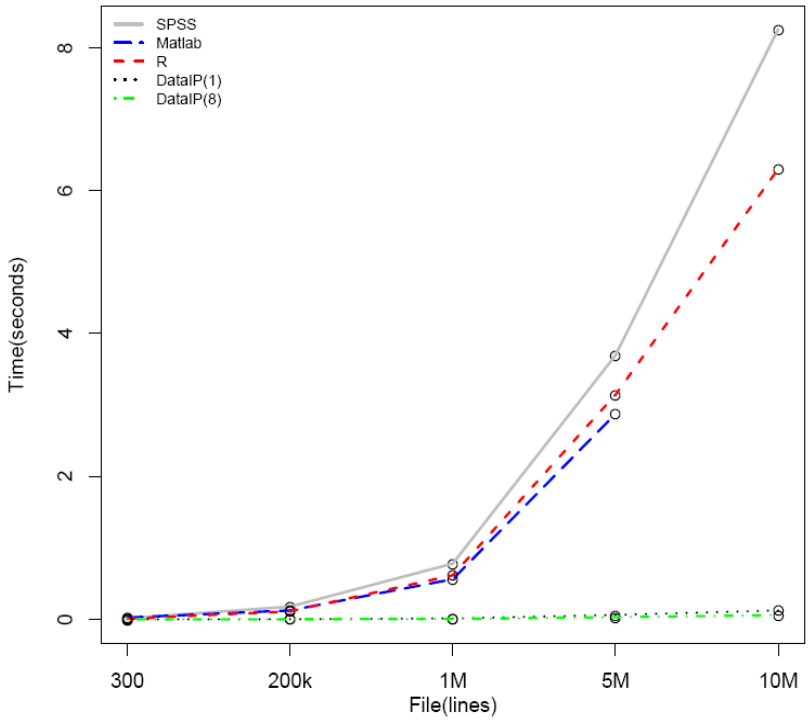


**Table 5** Pearson Correlation (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.016465374 | 0.177693620 | 0.779819110 |
| MATLAB | 0.024677000 | 0.124922000 | 0.561916000 |
| R | 0.006066667 | 0.113100000 | 0.617166700 |
| DataIP (1) | 0.000030952 | 0.001865020 | 0.013641000 |
| DataIP (8) | 0.000230371 | 0.001433520 | 0.006151690 |
| | *5 M* | | *10 M* |
| SPSS | 3.694809988 | | 8.257494096 |
| MATLAB | 2.874032000 | | - |
| R | 3.143700000 | | 6.302133000 |
| DataIP (1) | 0.062067200 | | 0.124550000 |
| DataIP (8) | 0.030884000 | | 0.059715300 |

For the Pearson correlation, all software increase execution times as the dataset size increases, with DataIP times growing linearly while SPSS, MATLAB and R grow exponentially. For smaller dataset sizes, R seems to perform better than SPSS and MATLAB. But as the dataset size increases, R execution times fluctuate behaving better than SPSS for 1 M and 5 M, but worse, for 10 M. For this experiment, DataIP manages to have modest speedups ranging from 1.3 to 2.2 for the larger datasets.

### *5.1.6 Spearman correlation*

For the Spearman correlation MATLAB and R start well with the smaller dataset size (300) having execution times better than SPSS. As the dataset sizes increase, R and MATLAB execution times become much worse than SPSS with MATLAB unable to run the largest 10 M dataset. DataIP manages to keep execution times low achieving a modest speedup for the larger datasets (1 M, 5 M and 10 M). Speedups for the larger datasets are limited, but in any case, the DataIP implementation largely outperforms MATLAB and R. Figure 8 shows execution times as we increase the dataset sizes.

**Table 6** Spearman correlation (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.935224150 | 2.222288790 | 5.096869300 |
| MatLab | 0.041846000 | 2.448189000 | 21.516139000 |
| R | 0.015900000 | 3.292867000 | 27.062666670 |
| DataIP (1) | 0.000106694 | 0.023312200 | 0.226401000 |
| DataIP (8) | 0.000460823 | 0.028749200 | 0.190727000 |
| | *5 M* | | *10 M* |
| SPSS | 17.836248300 | | 33.648206037 |
| MatLab | 85.345645000 | | - |
| R | 225.244500000 | | 526.125000000 |
| DataIP (1) | 1.528510000 | | 3.053720000 |
| DataIP (8) | 1.022340000 | | 2.031240000 |

**Table 7** Kendall correlation (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.028889376 | 1992.953648900 | - |
| MatLab | 0.146417000 | 4931.319953000 | - |
| R | 0.200633333 | - | - |
| DataIP (1) | 0.000356516 | 0.089785400 | 0.540615000 |
| DataIP (8) | 0.000442839 | 0.042243000 | 0.256585000 |
| | *5 M* | | *10 M* |
| SPSS | - | | - |
| MatLab | - | | - |
| R | - | | |
| DataIP (1) | 3.308700000 | | 6.887990000 |
| DataIP (8) | 1.516380000 | | 3.213720000 |

### *5.1.7 Kendall correlation*

The Kendall correlation seems to be the most inefficient implementation in all software, except in DataIP. SPSS starts well ahead of MATLAB and R for the 300 and 200 k dataset sizes, but fails to calculate the correlation for larger dataset sizes (1 M, 5 M and 10 M). On the other hand, DataIP runs for all dataset sizes with acceptable sequential execution times, and running two times faster when using 8 threads. For all dataset sizes, we managed to have an average speedup of 2.5 with 8 threads related to the single threaded version. MATLAB took almost 1.5 hour to run the 200 k dataset. Runs for MATLAB and R were aborted due to our imposed time limit. Implementations of Kendall usually employ the quadratic algorithm, but we use an $\mathcal{O}(n \log n)$ complexity version published many years ago (Knight, 1966). This can partially explain the poor performance achieved by MATLAB and R.

### *5.1.8 Significance tests*

Significance tests were ran only for the 300 dataset size. Table 8 shows average execution times for these tests. MATLAB does not provide functions to perform them.

**Table 8** Significance tests (time in seconds)

| | *Pearson* | *Spearman* | *Kendall* |
|---|---|---|---|
| R | 0.002900000 | 0.002900000 | 0.012600000 |
| DataIP (1) | 0.000006581 | 0.000006548 | 0.000007339 |
| DataIP (8) | 0.000294113 | 0.000345177 | 0.000321323 |

R performs quite well for these tests. However, DataIP can take advantage of code optimisations and runs more than 400 times faster than R to perform the Pearson and Spearman significance tests and more than 1,700 times faster than R to perform the Kendall significance test.

**Figure 8** Spearman correlation (see online version for colours)

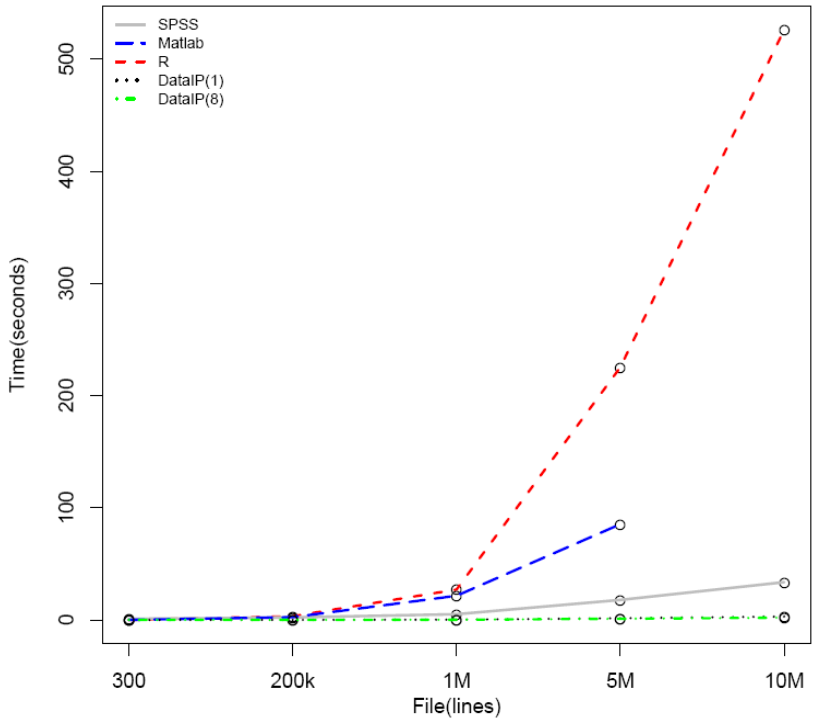
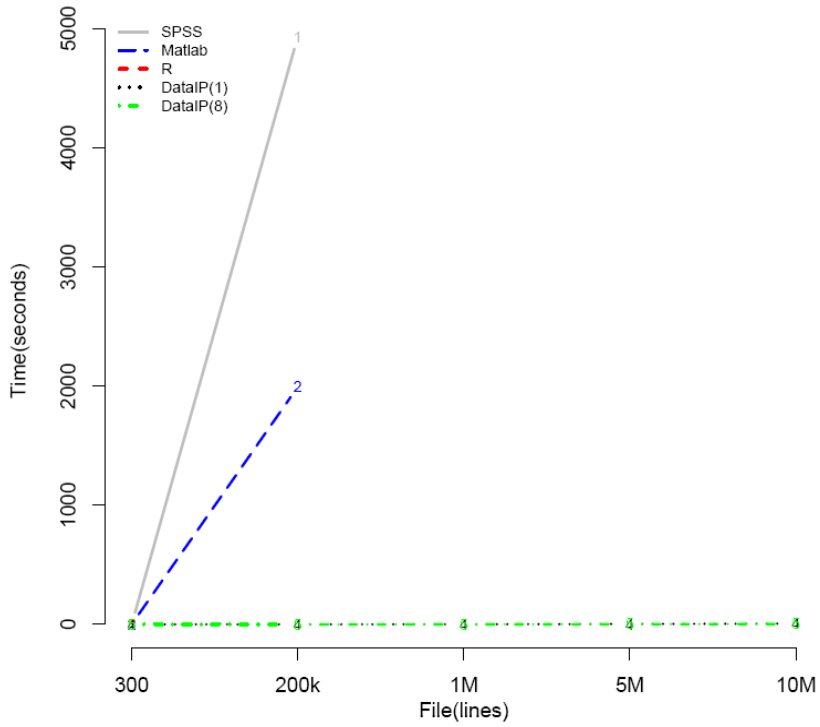



**Figure 9** Kendall correlation (see online version for colours)

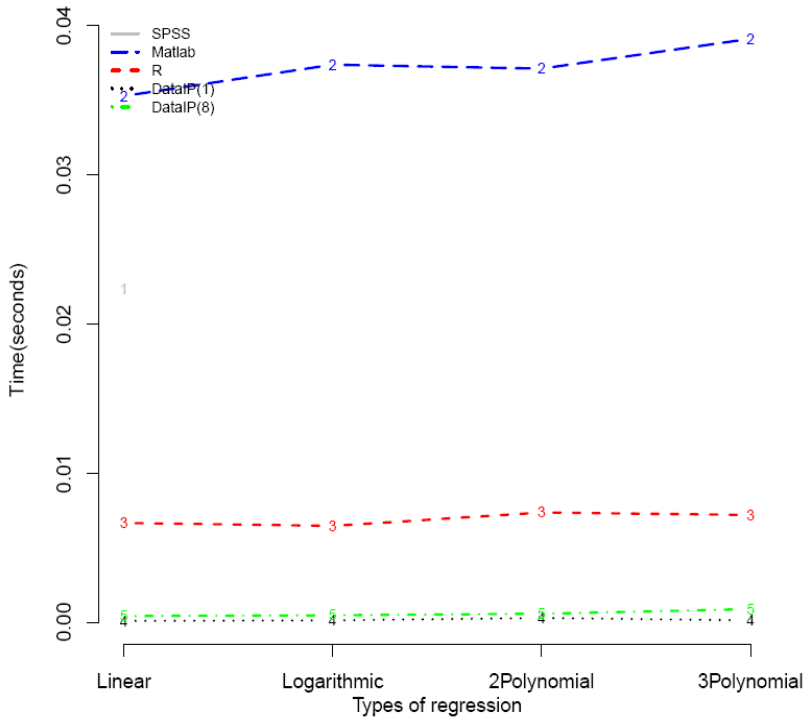



### *5.1.9 Regression*

Figure 10 shows the behaviour of all regressions for the 300 dataset.

Table 9 shows results of regression calculations for the 300 dataset. We performed linear, logarithmic, second and third order polynomial regressions.

**Figure 10** Regression (see online version for colours)




Average execution times for all types of regression calculation are very close for all software. R performs better than SPSS and MATLAB. DataIP, once more, wins by running the linear regression 234 times faster than SPSS, 372 times faster than MATLAB and 70 times faster than R. Speeds are quite similar when comparing with the other regressions.

Unfortunately, DataIP does not have any speedup when running with 8 threads. The reason for that is the very low execution time for one thread. Using 8 threads causes too much overhead because there is not enough work to keep all threads busy.

**Table 9** Regression (time in seconds)

| | *Linear* | *Logarithmic* |
|---|---|---|
| SPSS | 0.022341967 | - |
| MATLAB | 0.035284000 | 0.037382000 |
| R | 0.006666667 | 0.006466667 |
| DataIP (1) | 0.000094677 | 0.000135548 |
| DataIP (8) | 0.000424032 | 0.000477548 |
| | *2nd order polynomial* | *3rd order polynomial* |
| SPSS | - | - |
| MATLAB | 0.037118000 | 0.039120000 |
| R | 0.007366667 | 0.007200000 |
| DataIP (1) | 0.000295774 | 0.000136823 |
| DataIP (8) | 0.000581113 | 0.000887871 |

### *5.1.10 Wilcoxon, K-S, T, ANOVA and Bartlett tests*

Table 10 shows the execution times for the five statistical tests we implemented: Wilcoxon, K-S, T, ANOVA and Bartlett.

Figure 11 shows the behaviour of all tests for the 300 dataset, for all software.

**Table 10** Wilcoxon, K-S, T, ANOVA and Bartlett tests (time in seconds)

| | *Wilcoxon* | *K-S* | *T* |
|---|---|---|---|
| SPSS | 0.074964336 | 0.075011739 | 0.031844190 |
| MATLAB | 0.000811000 | 0.000362000 | 0.001214000 |
| R | 0.006633333 | 0.003166667 | 0.002266667 |
| DataIP (1) | 0.000153903 | 0.000007500 | 0.000005903 |
| DataIP (8) | 0.000439532 | 0.000437113 | 0.000448258 |
| | *ANOVA* | *Bartlett* | |
| SPSS | 0.022341967 | 0.031525859 | |
| MATLAB | 0.128183000 | - | |
| R | 0.006300000 | 0.000733333 | |
| DataIP (1) | 0.000011113 | 0.000003016 | |
| DataIP (8) | 0.000447371 | 0.000001710 | |

We ran the Bartlett test only in SPSS, R and DataIP, because the Bartlett function of MATLAB gives a result that is very much different than R, SPSS or our implementation. It looks like it does something more than just performing the test. MATLAB has a poor performance

for ANOVA when compared with SPSS, R or DataIP. SPSS seems to be the least efficient to run these tests.

#### *5.1.11 Principal component analysis*

Our last set of experiments for Machine 1 is to perform PCA. Table 11 shows average execution times for all datasets and all software.

SPSS is the most efficient to execute PCA. R performs quite poorly, and DataIP runs a bit more than two times slower than SPSS. PCA is not yet optimised in DataIP, therefore its sequential and parallel versions still need to be improved.

**Table 11** PCA (time in seconds)

| | *300* | *200 k* | *1 M* |
|---|---|---|---|
| SPSS | 0.024010287 | 0.193039497 | 0.843310515 |
| R | 0.017866667 | 1.537466700 | 7.165300000 |
| DataIP (1) | 0.002665940 | 0.301419000 | 1.536630000 |
| DataIP (8) | 0.002892400 | 0.301174000 | 1.514880000 |
| | *5 M* | *10 M* | |
| SPSS | 3.877115753 | 7.727483537 | |
| R | 34.187100000 | 71.544770000 | |
| DataIP (1) | 7.599020000 | 16.948700000 | |
| DataIP (8) | 7.307220000 | 16.495300000 | |

### *5.2 Dataset 2*

#### *5.2.1 Kendall correlations performance on big data*

In this section, we present results for the comparison between the same software running on Linux and Windows, using a larger number of instances (Dataset 2), on Machine 2. The dataset has 300 million instances. The tests are run on a single thread, since the dataset has only two variables and we parallelise the column operations.

**Table 12** Kendall correlations with DataIP (time in seconds)

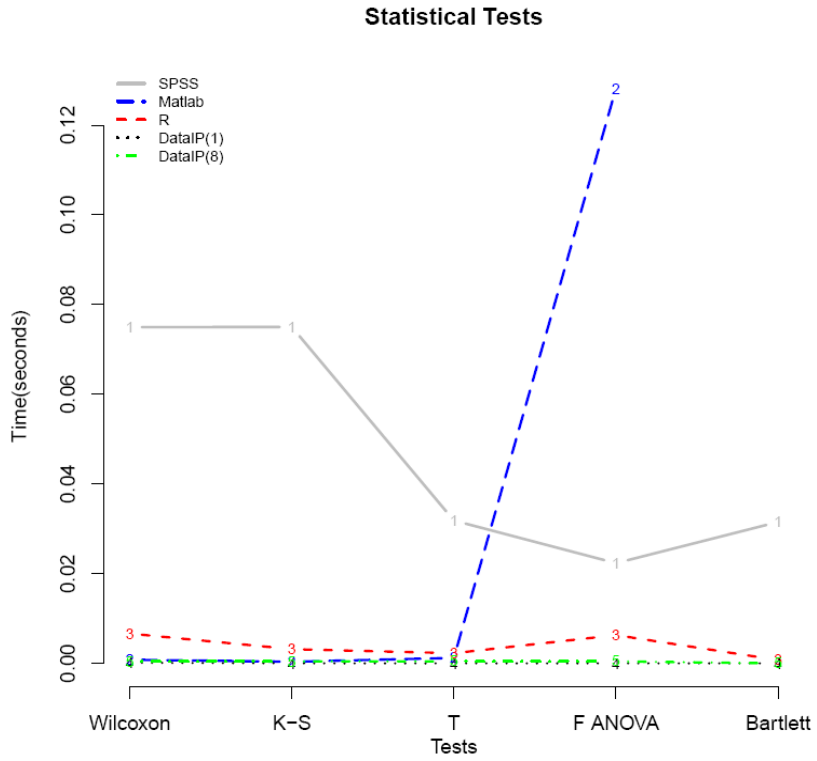
| | *300 M* |
|---|---|
| Windows | 116.489 |
| Linux | 113.264 |

Linux is slightly faster than Windows. In both environments, our implementation performs well running the Kendall correlation in less than 2 minutes. Unfortunately, we could not load the dataset in R or MatLab.

#### *5.2.2 Kendall correlations on Oracle Database 12c enterprise*

In this section we compare DataIP with the Oracle implementation of Kendall, in Machine 3, the Windows Server 2012 Datacenter using Dataset 2.

**Figure 11** Wilcoxon, K-S, T, ANOVA and Bartlett tests



The Oracle implementation has time complexity that most probably is quadratic (given its behaviour as we increase the input size). In order to perform this experiment, we had to reduce our 300 million instances. We created two smaller subsets: one with about 50 thousand instances and another one with about 200 thousand instances.

**Table 13** Kendall correlation (time in seconds)

| | *50 k* | *200 k* |
|---|---|---|
| Oracle 12c Enterprise | 426.46 | 6,253.568 |
| DataIP (1) | 0.015 | 0.046 |

Table 13 shows the results for this experiment. To get exactly the same results, on the 50 k subset, DataIP is 28,430.7 times faster than Oracle 12c Enterprise, and on the 200 k subset is 135947.1 times faster than Oracle 12c Enterprise. With larger datasets, we would even better results, since on the Oracle 12c Enterprise the execution time grows faster than on DataIP.

### *5.3 Dataset 3 and Dataset 4*

In this section, we evaluate DataIP, Matlab and R on data taken from the UCI Machine Learning Repository, with larger numbers of variables: Covertype and YearPredictionMSD (Dataset 3 with 55 variables and 581,012 instances, and Dataset 4 with 91 variables and 515,345 instances, respectively). Table 14 shows execution times in seconds averaged from 30 runs for all tests applicable to these datasets, using R, MATLAB and DataIP.

Regarding data input, R is the slowest, while DataIP using 8 threads, competitive with MATLAB, is the fastest to load the two datasets. To perform the summary, DataIP (sequential or using 8 threads) is several times faster than MATLAB or R for both datasets, thanks to its initial preprocessing. Calculating chi-square is also faster using DataIP (we did not run this for MATLAB for the same reasons given in previous sections: MATLAB's `chi2gof` does not produce the same results as R and DataIP). With respect to the calculation of correlations, DataIP remains the best, by several orders of magnitude. We also used R's `cor.fk` to calculate the Kendall correlation for both datasets. Times are reported for the correlation between all pairs of variables. R's `cor.fk` is also implemented using Knight's algorithm, which has $\mathcal{O}(n \log n)$ complexity. R ordinary Kendall implementation and MATLAB's could not finish before 15 hours of computation for both datasets. Spearman could not run in less than 15 hours in MATLAB and could not finish before 15 hours, in R, for the larger number of variables (91). Pearson, in R, for the largest dataset (Dataset 4) could not finish the 30 times replicated function to calculate the correlation. We then decided to report on just one run of Pearson for all pairs of variables, using R. Also, when loading data, R could not finish replicating 30 times the loading of Dataset 4. In this case, we also decided to report time for loading the file just once.

Shapiro-Wilk was not applied to these datasets, because they have more than 5,000 instances. Significance tests and regressions were also not calculated because no pair of variables had all three correlations strong (all of them greater than 0.75).

**Table 14** Covertype and YearPredictionMSD (time in seconds)

| *Function* | *System* | *Covertype* | *YearPred* |
|---|---|---|---|
| Data input | MATLAB | 5.153896000 | 11.599155000 |
| | R | 11.239400000 | 759.303000000 |
| | DataIP (1) | 14.915800000 | 50.017300000 |
| | DataIP (8) | 5.106310000 | 11.509100000 |
| Summary | MATLAB | 11.509100000 | 19.491427000 |
| | R | 22.150366670 | 98.33860000 |
| | DataIP (1) | 0.613753000 | 1.330550000 |
| | DataIP (8) | 0.993580000 | 2.064320000 |
| Chi-square | R | 7.939400000 | 25.005630000 |
| | DataIP (1) | 0.028485300 | 0.043176500 |
| | DataIP (8) | 0.028518900 | 0.043124200 |
| Pearson correlation | MATLAB | 17.337545000 | 42.161247000 |
| | R | 200.327500000 | 1,672.516000000 |
| | DataIP (1) | 1.190970000 | 2.716040000 |
| | DataIP (8) | 0.719797000 | 1.755250000 |
| Spearman correlation | MATLAB | (Aborted) | (Aborted) |
| | R | 798.738400000 | (Aborted) |
| | DataIP (1) | 15.489100000 | 67.36650000 |
| | DataIP (8) | 9.827960000 | 39.189800000 |
| Kendall correlation | MATLAB | (Aborted) | (Aborted) |
| | R | (Aborted) | (Aborted) |
| | R (cor.fk) | 788.131000000 | 4,230.495000000 |
| | DataIP (1) | 50.919000000 | 266.753000000 |
| | DataIP (8) | 16.688200000 | 51.050800000 |

## 6 Conclusions

Our main conclusion is that traditional implementations of basic statistical functions, crucial for data analysis, need to be revisited, and better designed to meet the requirements of larger datasets. We presented results of a C/C++ implementation of many statistical functions and show that, even for small datasets, well designed code can achieve very good performance when compared with state-of-the-art statistical software. Experiments running with 1 thread or 8 threads perform several orders of magnitude faster than SPSS, R or MATLAB. Besides, we can achieve reasonable speedups taking advantage of a multicore machine, which MATLAB and R can also take, but, even with perfect speedups, would not beat DataIP. DataIP was the only software that completed all experiments while the other software failed some experiments due to memory or timing issues.

## Acknowledgements

We are grateful to Professor Domingos Alves, from Faculty of Medicine of Ribeirão Preto, São Paulo, Brazil, who kindly provided us with the patient discharge dataset. We would also like to thank the anonymous referees that helped improving the quality of this work. This work was supported by NLPC, Lda (Proj. DATAIP Nb. 38667, 07/2012-SII&DT), IAPMEI/COMPETE/QREN/ EUROPEAN UNION, Copyright ©2014 2015 NLPC – I.C. NLPC T.A.I.C.C. Gestão, LDA – All rights reserved. Other funding was provided by the European Regional Development Fund as part of project NanoSTIMA (NORTE-01-0145-FEDER-000016).

DataIP is a R&D project from NLPC – Dataias. All contacts related to this work should be addressed to: rd@dataias.com, http://www.dataias.com.

## References

Casella, G. and Berger, R. (2008) *Statistical Inference*, Duxbury Advanced Series, Duxbury Thomson Learning, Pacific Grove.

Chambers, J.M., Freeny, A. and Heiberger, R.M. (1992) *Analysis of Variance; Designed Experiments*, Wadsworth & Brooks/Cole, Pacific Grove.

Chandra, R. (2001) *Parallel Programming in OpenMP, High Performance Computing*, Morgan Kaufmann Publishers, San Diego.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.E. (2008) 'Bigtable: a distributed storage system for structured data', *ACM Trans. Comput. Syst*., Vol. 26, No. 2, pp.4:1–4:26.

Christensen, D. (2005) 'Fast algorithms for the calculation of Kendall's τ', *Computational Statistics*, Vol. 20, No. 1, pp.51–62.

Conover, W. (2006) *Practical Nonparametric Statistics*, Cram101 Series, Cram101 Incorporated, New York.

Croux, C. and Dehon, C. (2010) 'Influence functions of the spearman and Kendall correlation measures', *Statistical Methods & Applications*, Vol. 19, No. 4, pp.497–515.

Hollander, M. and Wolfe, D.A. (1973) *Parallel Programming in OpenMP. Nonparametric Statistical Methods*, John Wiley & Sons, Kendall and Spearman Tests, New York.

Hollander, M., Wolfe, D. and Chicken, E. (2013) *Nonparametric Statistical Methods*, Wiley Series in Probability and Statistics, Wiley, New York.

Hsu, C-H., Slagter, K.D. and Chung, Y-C. (2015) 'Locality and loading aware virtual machine mapping techniques for optimizing communications in mapreduce applications', *Future Gener. Comput. Syst*., Vol. 53, No. C, pp.43–54.

Kane, M., Emerson, J. and Weston, S. (2013) 'Scalable strategies for computing with massive data', *Journal of Statistical Software*, Vol. 55, No. 1, pp.1–19.

Kendall, M. and Gibbons, J. (1990) *Rank Correlation Methods*, A Charles Griffin Title, Edward Arnold, London.

Knight, W.R. (1966) 'A computer method for calculating Kendall's tau with ungrouped data', *Journal of the American Statistical Association*, Vol. 61, No. 314, pp.436–439.

Marsaglia, G., Tsang, W.W. and Wang, J. (2003) 'Evaluating Kolmogorov's distribution', *Journal of Statistical Software*, Vol. 8, No. 18, pp.1–4.

Montgomery, D. and Runger, G. (2010) *Applied Statistics and Probability for Engineers*, John Wiley & Sons, New York.

Press, W. (2007) *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, Cambridge.

Royston, P. (1995) 'Remark as r94: a remark on algorithm as 181: the w-test for normality', *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, Vol. 44, No. 4, pp.547–551.

Rumsey, D. (2010) *Statistics Essentials for Dummies*, For Dummies, Wiley, Hoboken.

Slagter, K., Hsu, C-H. and Chung, Y-C. (2015) 'An adaptive and memory efficient sampling mechanism for partitioning in mapreduce', *International Journal of Parallel Programming*, Vol. 43, No. 3, pp.489–507.

Slagter, K., Hsu, C-H., Chung, Y-C. and Zhang, D. (2013) 'An improved partitioning mechanism for optimizing massive data analysis using mapreduce', *The Journal of Supercomputing*, Vol. 66, No. 1, pp.539–555.

Venables, W.N. and Ripley, B.D. (2002) *Modern Applied Statistics with S*, Springer-Verlag, New York.

Wonnacott, T. and Wonnacott, R. (1990) *Student Workbook, Introductory Statistics for Business and Economics, Fourth Edition and Introductory Statistics*, 5th ed., Wiley, New York.

Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I. (2010) 'Spark: cluster computing with working sets', in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10*, USENIX Association, Berkeley, CA, USA, p.10.