Time-series Bitmaps: a Practical Visualization Tool for Working with Large Time Series Databases

Nitin Kumar Venkata Nishanth Lolla Eamonn Keogh Stefano Lonardi Chotirat Ann Ratanamahatana Li Wei
University of California - Riverside
Department of Computer Science & Engineering
Riverside, CA 92521, USA
{nkumar, vlolla, eamonn, stelo, ratana, wli}@cs.ucr.edu

Abstract

The increasing interest in time series data mining in the last decade has resulted in the introduction of a variety of similarity measures, representations, and algorithms. Surprisingly, this massive research effort has had little impact on real world applications. Real world practitioners who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our system has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Microsoft Windows, thus making deployment easier. Our approach extracts features from a time series of arbitrary length and uses information about the relative frequency of its features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection. We demonstrate the utility of our approach with a set of comprehensive experiments on real datasets from a variety of domains.

Keywords: Time Series, Chaos Game, Visualization.

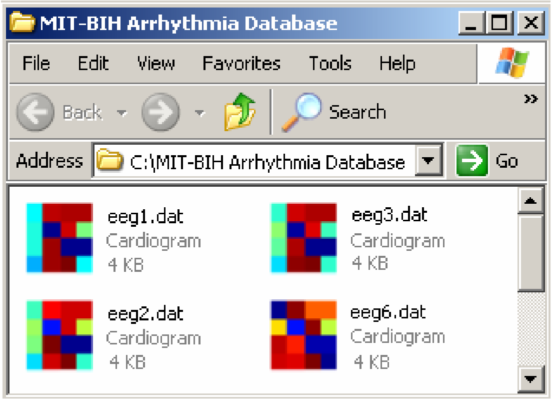
1 Introduction

The increasing interest in time series data mining in the last decade has resulted the introduction of a variety of similarity measures/ representations/ definitions/ indexing techniques and algorithms (see, e.g., [1][2][3][4] [9][13][14]). Surprisingly, this massive research effort has had little impact on real world applications. Examples of implemented systems are rare exceptions [17]. Cardiologists, engineers, technicians, and others who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. While it is difficult to firmly establish the reasons for the discrepancy between tool availability and practical adoption, the following reasons suggested themselves after an informal survey.

- Time series data mining tools often come with a bewildering number of parameters. It is not obvious to the practitioner how these should be set [15].
- Research tools often require (relatively) specialized hardware and/or software, rather than the ubiquitous desktop PC/Windows environment that prevails.
- Many tools have a steep learning curve, requiring the user to spend many unproductive hours learning the system before any possibility of useful work.

In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our approach extracts features from a time series of arbitrary length, and uses information about the relative frequency of these features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of these bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection.

While our system can be used as an interactive tool, it also has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Windows, Aqua, X-windows, etc. Since users navigate through files by looking at their icons, we decided to employ the bitmap representation as the icon corresponding to each time series. Simply by glancing at the icons contained in a folder of time series files, a user can quickly identify files that require further investigation. In Figure 1, we have illustrated a simple example.



C:\MIT-BIH Arrhythmia Database

eeg1.dat
Cardiogram
4 KB

eeg3.dat
Cardiogram
4 KB

eeg2.dat
Cardiogram
4 KB

eeg6.dat
Cardiogram
4 KB

Figure 1. Four time series files represented as time series thumbnails. While they are all examples of congestive heart failure, *eeg6.dat* is from a different person to all the rest. This fact is immediately apparent from a casual inspection of the thumbnail representation.

The utility of the idea shown in Figure 1 can be further enhanced by arranging the icons within the folder by *pattern similarity*, rather than the typical choices of arranging them by *size*, *name*, *date*, etc. This can be achieved by using a simple multidimensional scaling or a self-organizing map algorithm to arrange the icons.

Unlike most visualization tools which can only be evaluated subjectively, we will perform objective evaluations on the amount of useful information contained within a time series bitmap. More precisely, we will analyze the loss of accuracy of classification /clustering/anomaly detection algorithms when the input is based solely on the information contained in the bitmap. As we will show, the experiments strongly confirm the utility of our approach.

2 Chaos Game Representations

Our visualization technique is partly inspired by an algorithm to draw fractals called the *Chaos game* [3]. The method can produce a representation of DNA sequences, in which both local and global patterns are displayed. For example, a biologist can recognize that a particular substring, say in a bacterial genome, is rarely used. This would suggest the possibility that the bacteria have evolved to avoid a particular restriction enzyme site, which means that he/she might not be easily attacked by a specific bacterio-phage.

From our point of view, the crucial observation is that the CGR representation of a sequence allows the investigation of the patterns in sequences, giving the human eye a possibility to recognize hidden structures.

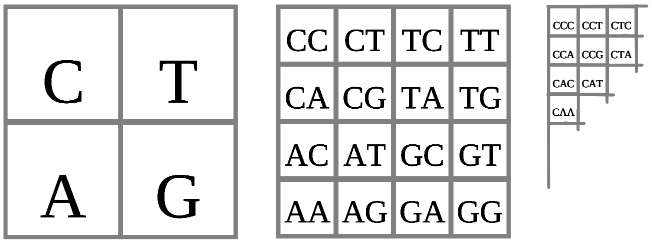
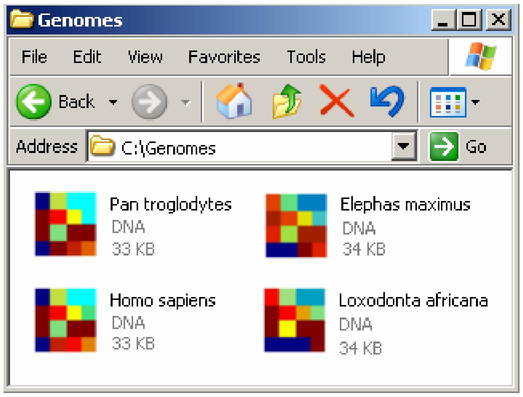


Figure 2. The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution

We can get a hint of the potential utility of the approach if, for example, we take the first 16,000 symbols of the mitochondrial DNA sequences of four familiar species and use them to create their own file icons. Figure 3 below illustrates this. Even if we did not know these particular animals, we would have no problem recognizing that there are two pairs of highly related species being considered.

With respect to the non-genetic sequences, Joel Jeffrey noted, *"The CGR algorithm produces a CGR for any sequence of letters"*[9]. However, it is only defined for discrete sequences, and most time series are real valued.

This encouraged us to try a similar technique on time series data and investigate the utility of such representation on the classic data mining tasks of clustering, classification, and visualization.



Pan troglodytes
DNA
33 KB

Elephas maximus
DNA
34 KB

Homo sapiens
DNA
33 KB

Loxodonta africana
DNA
34 KB

Figure 3. The gene sequences of mitochondrial DNA of four animals, used to create their own file icons using a chaos game representation. Note that *Pan troglodytes* is the familiar Chimpanzee, and *Loxodonta africana* and *Elephas maximus* are the African and Indian Elephants, respectively. The file icons show that humans and chimpanzees have similar genomes, as do the African and Indian elephants.

Since CGR involves treating a data input as an abstract string of symbols, a discretization method is necessary to transform continuous time series data into discrete domain. For this purpose, we used the Symbolic Aggregate approXimation (SAX) [18]. While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols [7], the SAX technique of Lin *et. al.* [18] is unique and ideally suited for data mining. SAX is the only symbolic representation that allows the lower bounding of the distances in the original space. The ability to efficiently lower bound distances is at the heart of hundreds of indexing algorithms and data mining techniques [2][6][12][14][18][19].

The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. By substituting each section with its mean, a reduced dimensionality piecewise constant approximation of the data is obtained. This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 4 shows a short time series being converted into the SAX word **baabccbc**.

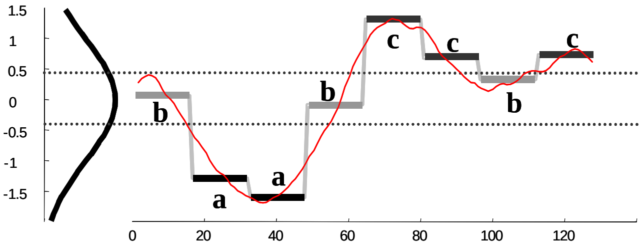


Figure 4. A real valued time series can be converted to the SAX word **baabccbc**. Note that all three possible symbols are approximately equally frequent.

The time and space complexity to convert a sequence to its SAX representation is linear in the length of the sequence. The SAX representation has been successfully used by various groups of researchers for indexing, classification, clustering [18], motif discovery [5][6][21], rule discovery, [20], visualization [17], and anomaly detection [15].

3 Time Series Bitmaps

At this point, the connection between the two "ingredients" for the time series bitmaps should be evident. We have seen in Section 2.3 that the *Chaos game* [3] bitmaps can be used to visualize discrete sequences, and we have seen in Section 2.4 that the SAX representation is a discrete time series representation that has demonstrated great utility for data mining. It is natural to consider combining these ideas.

The *Chaos game* bitmaps are defined for sequences with an alphabet size of four. It is fortuitous that DNA strings have this cardinality. SAX can produce strings on any alphabet sizes. As it happens, a cardinality of four (or three) has been reported by many authors as an excellent choice for diverse datasets on assorted problems [5][6][15][17][18] [20][21].

We need to define an initial ordering for the four SAX symbols **a**, **b**, **c**, and **d**. We use simple alphabetical ordering as shown in Figure 5.

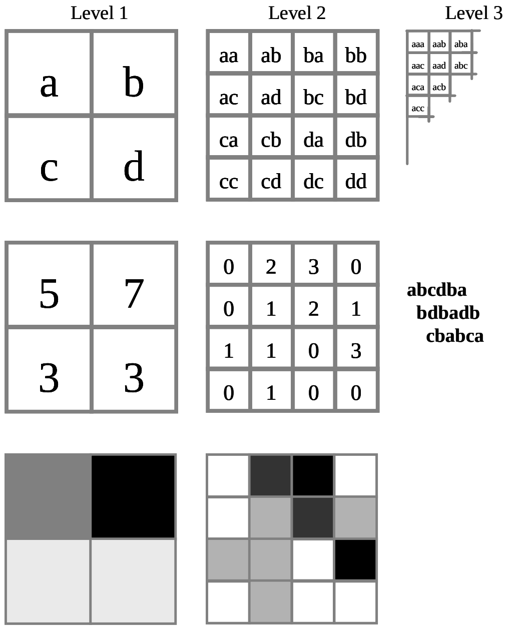


Figure 5. *Top*) The four possible SAX symbols are mapped to four quadrants of a square, and pairs, triplets, etc are recursively mapped to finer grids. *Middle*) We can extract counts of symbols from a SAX representation and record them in the grids. *Bottom*) The recorded values can be linearly mapped to colors, thus creating a square bitmap.

After converting the original raw time series into the SAX representation, we can count the frequencies of SAX "subwords" of length *L*, where *L* is the desired level of recursion. Level-1 frequencies are simply the raw counts of the four symbols. For level 2, we count pairs of subwords of size 2 (**aa**, **ab**, **ac**, etc). Note that we only count subwords taken from individual SAX words. For example, in the SAX representation in Figure 5 *middle right*, the last symbol of the first line is **a**, and the first symbol of the second word is **b**. However, we do not count this as an occurrence of **ab**.

Once the raw counts of all subwords of the desired length have been obtained and recorded in the corresponding pixel of the grid, a final step is required. Since the time series in a data collection may be of various lengths, we normalize the frequencies by dividing it by the largest value. The pixel values thus range from 0 to 1. The final step is to map these values to colors. In the example above, we mapped to grayscale, with 0 = *white*, 1 = *black*. However, it is generally recognized that grayscale is not *perceptually uniform* [22]. A color space is said to be perceptually uniform if small changes to a pixel value are approximately equally perceptible across the range of that value. For all images produced in this paper we use Matlab's "jet" color space, which is a linearization of a large fraction of all possible colors and which is designed to be perceptually uniform.

Note that unlike the arbitrarily long, and arbitrarily shaped time series from which they where derived, for a fixed *L*, the bitmaps have a constant space and structure.

We do not suggest any utility in viewing a single time series bitmap. The representation is abstract, and we do not expect a user to be able to imagine the structure of time series given the bitmap. The utility of the bitmaps comes from the ability to efficiently compare and contrast them.

4 Time Series Thumbnails

A unique advantage of the time series bitmap representation is the fact that we can transparently integrate it into the user graphical interface of most standard operating systems. Since most operating systems use the ubiquitous square icon to represent a file, we can arrange for the icons for time series files to appear as their bitmap representations. Simply by glancing at the contents of a folder of time series files, a user may spot files that require further investigation, or note natural clusters in the data.

The largest possible icon size varies by operating system. All modern versions of Microsoft Windows support 32 by 32 pixels, which is large enough to support a bitmap of level 5. As we will see, level 2 or 3 seems adequate for most tasks/datasets. To augment the utility of the time series bitmaps, we can arrange for their placement on screen to reflect their structure. Normally, file icons are arranged by one of a handful of common criteria, such as *name*, *date*, *size*, etc.

We have created a simple modification of the standard Microsoft Windows (98 or later) file browser by introducing the concept of *Cluster View*. If *Cluster View* is chosen by the user, the time series thumbnails arrange themselves by similarly. This is achieved by performing Multi-Dimensional Scaling (MDS) of the bitmaps, and projecting them into a 2 dimensional space. For aesthetic reasons, we "snap" the icons to the closest grid point.

Figure 6 displays an example of *Cluster View* in Microsoft Windows XP Operating System. In this example, the *Cluster View* is obtained for five MIT-BIH Arrhythmia Database files. It is evident in the figure that *eeg1.dat*, *eeg2.dat*, and *eeg3.dat* belong to one cluster whereas *eeg6.dat* and *eeg7.dat* belong to another. In this case, the grouping correctly reflects the fact that latter two files come from a different patient to first three.

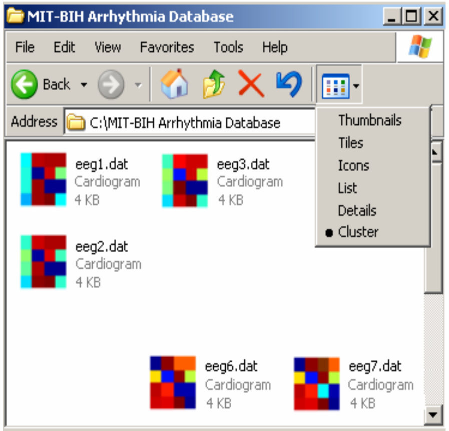


Figure 6. A snapshot of a folder containing cardiograms when its files are arranged by "Cluster" option. Five cardiograms have been grouped into two different clusters based on their similarity.

5 Experimental Evaluation

We have tested our proposed approach with a comprehensive set of experiments; most of these were omitted for the sake of brevity. We urge the interested reader to consult the full version of this paper or URL [11] for large-scale color reproductions and additional details.

We examined the UCR Time Series Archive for datasets that come in pairs. For example, in the **Buoy Sensor** dataset, there are two time series, *North Salinity* and *East Salinity*, and in the **Exchange Rate** dataset, there are two time series, *German Marc* and *Swiss Franc*. We were able to identify fifteen such pairs, from a diverse collection of time series covering the domains of finance, science, medicine, industry, etc. Although our method is able to deal with time series of different lengths, we truncated all time series to length 1,000 for visual clarity.

While the correct hierarchical clustering at the top of the tree is somewhat subjective, at the lower level of the tree, we would hope to find a single bifurcation separating each pair in the dataset. Our metric, Q, for the quality of clustering is therefore the number of such correct bifurcations divided by fifteen, the number of datasets. For a perfect clustering, $Q = 1$. Figure 7 shows the resulting dendrogram for our approach. The dendrograms for the other approaches are omitted here for brevity, but may be viewed at [11].

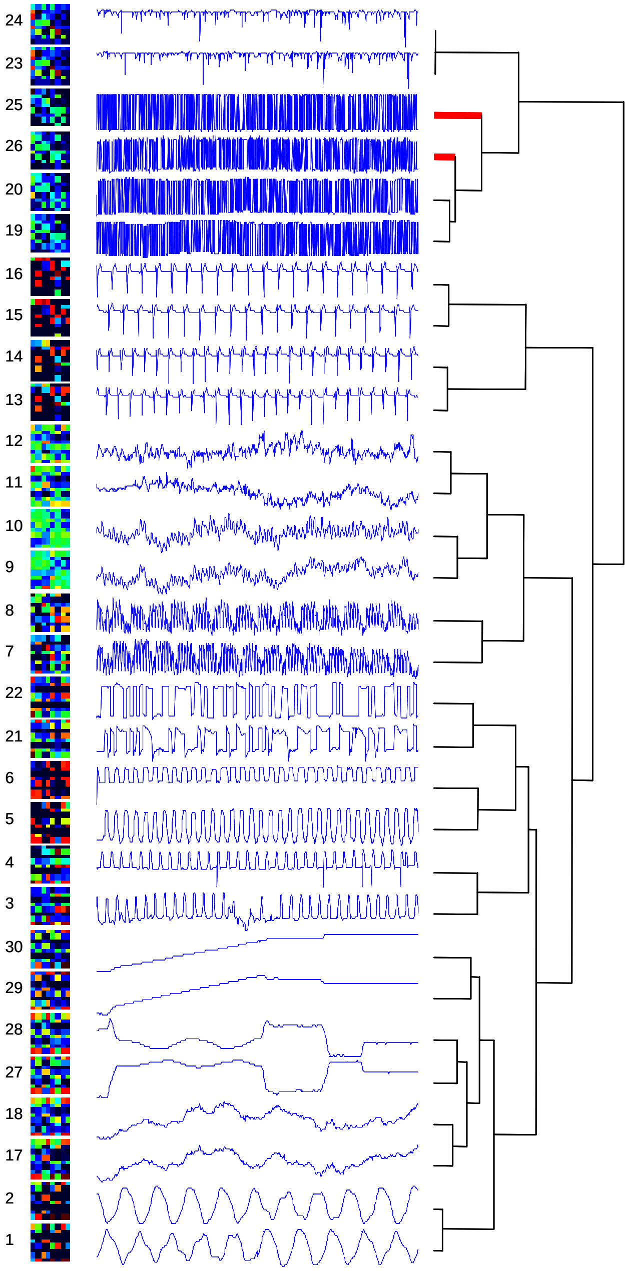


Figure 7. The clustering obtained by the time series thumbnail approach on a heterogeneous data collection. Bold lines denote incorrect subtrees. A key of the data appears in Appendix A of the full paper.

We compared to two well-known and highly referenced techniques, Markov models [8] and ARIMA models [10][22]. For each technique, we spent one hour searching over parameter choice and reported only the best performing result. To mitigate the problem of overfitting, we set the parameters on a different, but similar dataset. The results for the three approaches are given in Table 1.

Table 1. The quality of clustering obtained by the three algorithms under consideration.

Algorithm	**Q**
Thumbnails	0.93
Markov Model	0.46
ARMA models	0.40

6 Conclusions and Future Work.

In this work, we have introduced a new framework for visualization of time series. Our approach is unique in that it can be directly embedded into any standard GUI operating system. We demonstrated the effectiveness of our approach on a variety of tasks and domains. Future work includes an extensive user study, and investigating techniques to automatically set the system parameters.

This research was partly funded by the National Science Foundation under grant IIS-0237918.

Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets used in this work are available at the following URL [11].

References

[1] Aach, J., & Church, G. (2001). *Aligning gene expression time series with time warping algorithms*. Bioinformatics, Volume 17, pp. 495-508.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). *Fast similarity search in the presence of noise, scaling, and translation in times-series databases*. In Proceedings of twenty-first International Conference on Very Large Databases, pp. 490-501.

[3] Barnsley, M.F., & Rising, H. (1993). *Fractals Everywhere*, second edition, Academic Press.

[4] Berndt, D., & Clifford, J. (1994). *Using dynamic time warping to find patterns in time series*, AAAI Workshop on Knowledge Discovery in Databases, pp. 229-248.

[5] Celly, B. & Zordan, V. B. (2004). *Animated People Textures*. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland.

[6] Chiu, B., Keogh, E., & Lonardi, S. (2003). *Probabilistic Discovery of Time Series Motifs*. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493-498.

[7] Daw, C. S., Finney, C.E.A., & Tracy, E.R.(2001). *Symbolic Analysis of Experimental Data*. Review of Scientific Instruments. (2002-07-22).

[8] Ge, X., & Smyth, P. (2000). *Deformable Markov model templates for time-series pattern matching*. In proceedings of the sixth ACM SIGKDD, pp. 81-90.

[9] Jeffrey, H.J. (1992). *Chaos Game Visualization of Sequences*. Comput. & Graphics 16, pp. 25-33.

[10] Kalpakis, K., Gada, D., & Puttagunta, V. (2001). *Distance Measures for Effective Clustering of ARIMA Time-Series*. In the Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 273-280.

[11] Keogh, E. www.cs.ucr.edu/~nkumar/SDM05.html

[12] Keogh, E. (2002). *Exact indexing of dynamic time warping*. In Proceedings of the twenty-eighth International Conference on Very Large Data Bases, pp. 406-417.

[13] Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra (2001). *Locally adaptive dimensionality reduction for indexing large time series databases*. In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 151-162.

[14] Keogh, E. & Kasetty, S. (2002). *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration*. In the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 102-111.

[15] Keogh, E., Lonardi, S., & Ratanamahatana, C. (2004). *Towards Parameter-Free Data Mining*. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[16] Korn, F., Jagadish, H., & Faloutsos, C. (1997). *Efficiently supporting ad hoc queries in large datasets of time sequences*. In Proceedings of SIGMOD, pp. 289-300.

[17] Lin, J., Keogh, E., Lonardi, S., Lankford, J.P. & Nystrom, D.M. (2004). *Visually Mining and Monitoring Massive Time Series*. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining..

[18] Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*. In proceedings of the eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

[19] Ratanamahatana, C.A., & Keogh, E. (2004). *Everything you know about Dynamic Time Warping is Wrong*. 3rd Workshop on Mining Temporal and Sequential Data, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[20] Silvent, A. S., Carbay, C., Carry, P. Y. & Dojat, M. (2003). *Data, Information and Knowledge for Medical Scenario Construction*. In proceedings of the Intelligent Data Analysis In Medicine and Pharmacology Workshop.

[21] Tanaka, Y. & Uehara, K. (2004). *Motif Discovery Algorithm from Motion Data*. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI).

[22] Wyszecki, G. (1982). Color science: Concepts and methods, quantitative data and formulae, 2nd edition.