TECHNICAL BRIEF

JSparklines: Making tabular proteomics data come alive

Harald Barsnes[1], Marc Vaudel[1] and Lennart Martens[2,3]

[1] Proteomics Unit, Department of Biomedicine, University of Bergen, Norway
[2] Department of Medical Protein Research, Ghent University, Ghent, Belgium
[3] Department of Biochemistry, Ghent University, Ghent, Belgium

Received: July 29, 2014
Revised: September 15, 2014
Accepted: November 20, 2014

Perhaps the most common way of presenting proteomics data, and indeed life sciences data in general, is by using some form of tabular data. And while tables can be very informative and contain lots of information, the format can be challenging to interpret visually. An elegant and efficient solution is to extend the textual and numerical information with an additional visual layer, referred to as sparklines, making it intuitive to draw inferences about the properties of the underlying data. We here present a free and open source Java library called JSparklines (http://jsparklines.googlecode.com) that allows straightforward addition of a substantial list of customizable sparklines to tabular data representations, and we show examples of how these sparklines greatly simplify the interpretation of the tabular data.

Keywords:
Bioinformatics / Data visualization / Open Source / Sparklines

The use of tables to present data is employed throughout science, from the periodic table of elements in chemistry to logarithmic tables in mathematics. In combination with correct row and header annotations, the reader is usually able to interpret the data without requiring additional information. Indeed, organizing numerical data in tables makes it easier to locate the largest or smallest value, or to find rows or columns that seem to stand out from the rest.

However, when size and/or complexity of the included data grow, it becomes increasingly difficult to visually locate such features. A possible solution is then to extend the textual and numerical information in the table with an additional visual layer, referred to as sparklines. The concept of sparklines was introduced in 1983 by Edward R. Tufte in his classic book on *The Visual Display of Quantitative Information*, as "intense, simple, word-sized graphics" [1]. Here, Tufte presented the idea of visualizing the numbers in a tabular lay-out rather than (or in addition to) simply listing them. It did however, take decades before the concept gained traction. Perhaps the main limiting factor was the fact that in 1983, most graphics were made either by hand or on poor resolution computer displays, making the addition of visual elements labor intensive or difficult to assess accurately.

Correspondence: Dr. Harald Barsnes, Proteomics Unit, Department of Biomedicine, University of Bergen, Jonas Liesvei 91, N-5009 Bergen, Norway
E-mail: harald.barsnes@biomed.uib.no
Fax: +47-55-58-63-60

Yet today the picture is very different; data tables and graphics are now created on high resolution computer displays that are literally designed for graphically displaying and interacting with increasingly large amounts of information. In other words, the required framework is now in place to take full advantage of the powerful idea of sparklines. To illustrate the growing popularity of sparklines, consider the inclusion of sparklines in recent versions of MS Excel, and the large variety of sparklines variants that now frequently appear in newspapers and on web pages.

Even though there are also some examples of sparklines in the scientific community [2–7], their use is not as frequent as it ought to be. Especially in the highthroughput "omics" fields in the life sciences, improving the ease with which acquired data can be interpreted by collaborators and colleagues should be an important goal [8]. We therefore created an open source and freely available Java library, called JSparklines (http://jsparklines.googlecode.com), that makes it straightforward to add a plethora of highly customizable sparklines to tabular datasets and describe here its application to proteomic results visualization. JSparklines is based on Java Swing and the open source JFreeChart (http://www.jfree.org/jfreechart) library, allowing for a rich set of highly customizable charts, and making it easy to extend the list of supported sparklines. The library has been fine-tuned and extended over several

Colour Online: See the article online to view Figs. 1 and 2 colour online only.

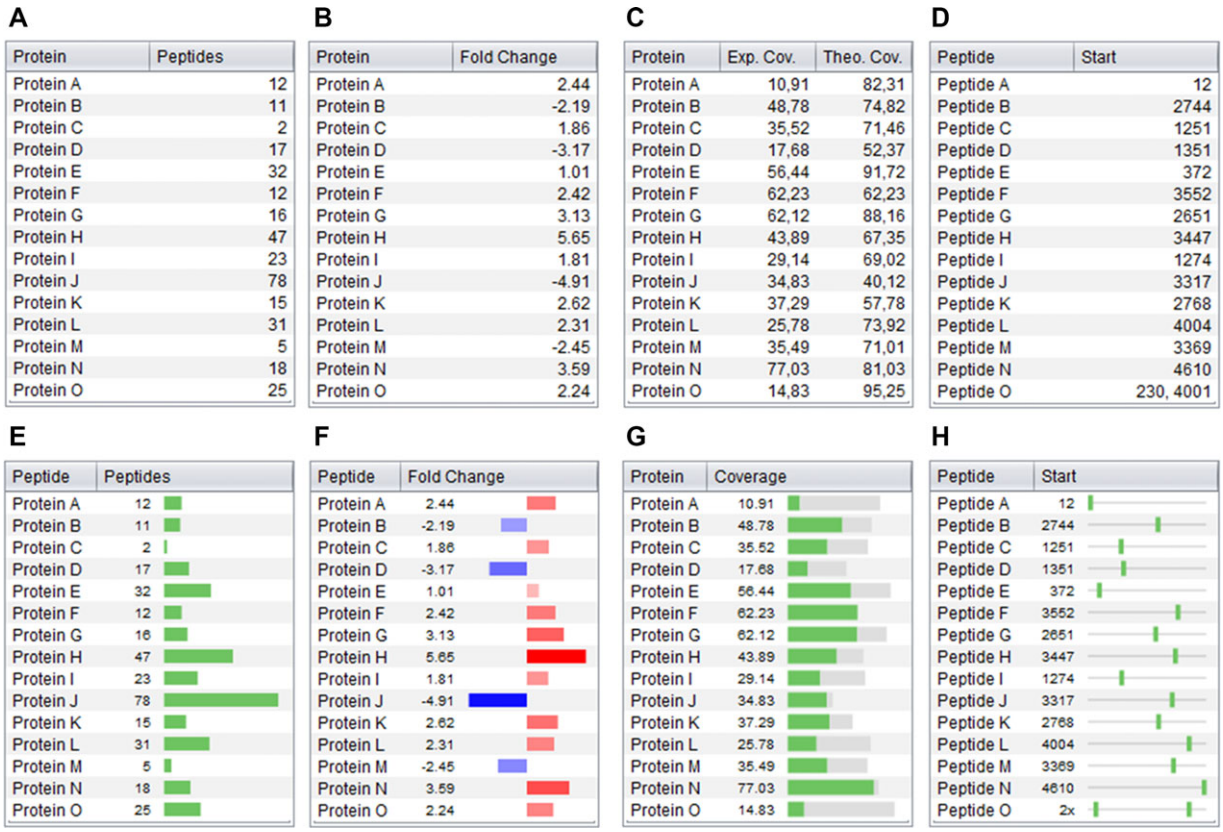

Figure 1. Examples of how JSparklines improves the visual display and guides the interpretation of tabular data: (A) and (E) show the number of peptides *per* protein; (B) and (F) provide quantitative protein data, up- and downregulations are illustrated in (F) using red and blue sparklines, respectively; (C) and (G) provide both theoretical (gray) and experimental (green) coverage of the protein sequences; and (D) and (H) provide information on the starting location of the peptides relative to the parent protein sequence, with (H) illustrating the relative location of the peptide in the protein sequence.

years and has already been applied in several bioinformatics applications [9–13].

Note that JSparklines expands on the original sparklines concept by including any type of cell-size graphic that can be used to improve the understanding of the underlying data. This includes bar charts, area charts, line charts, box plots, and scatter plots, along with simpler options such as equally sized, colored boxes to represent groups or categories.

JSparklines has been designed to make it as easy as possible to include sparklines into (existing) Java applications, requiring only the addition of a couple of lines of Java code to upgrade new or existing tables with sparklines. Indeed, when using single-value sparklines, no changes are needed for the data table itself, as JSparklines simply works as a visualization layer on top of the existing values.

The main concept employed by JSparklines is the use of so-called table cell renderers. A table cell renderer is responsible for how the content in a cell in a table is visualized, e.g., show the value as text or by some graphical representation of that value. Thus the same table content can be displayed in different ways simply by changing the table cell renderer for a given column. For example, a column of protein coverage values can be displayed as bar charts using the JSparklinesBarChartTableCellRenderer, or a column with multiple coverage values per protein, i.e., multiple values per cell, can be displayed as bar charts, line charts or area charts, etc., using the more generic JSparklinesTableCellRenderer.

For an overview of all the available table cell renderers and the input supported, see the JavaDoc documentation found at the JSparklines webpage, which also contains further details as the "How to use JSparklines" Wiki, plus two executable demos.

Figure 1 provides a simple example of the use of JSparklines to visualize data obtained from a typical quantitative proteomics experiment: number of peptides, fold changes, protein coverage, and peptide localization on the protein sequence. Note how it becomes much easier to isolate the protein with the most peptides by using the sparklines display instead of the table (Fig. 1A compared to Fig. 1E). Furthermore, when considering the table of up- and down-regulated protein expression values (Fig. 1B), the sparklines (Fig. 1F) make it much easier to find the most regulated proteins, in addition to immediately visually separating the up- and down-regulated proteins through the blue and red color coding.

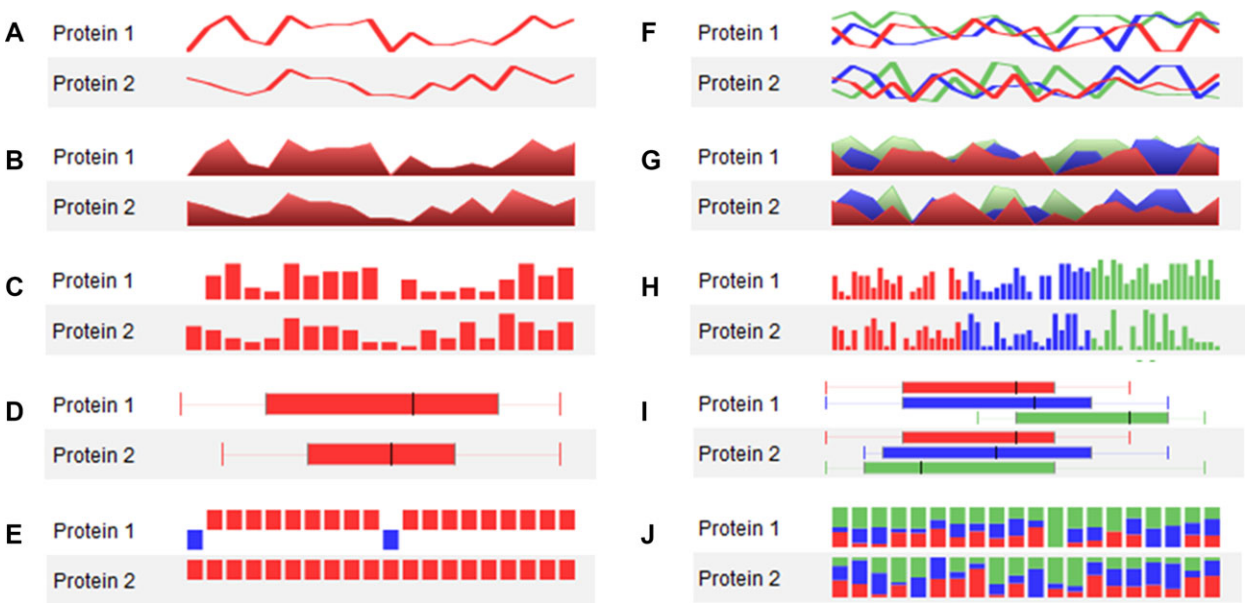

Figure 2. Examples of JSparklines showing multiple values per cell, using (simulated) spectrum counting results for 20 peptides for two protein isoforms; and multiple data sets, with multiple values each, per cell, using the same (simulated) spectrum counting comparing three different experimental setups. Multiple values per cell are represented as (A) line charts, (B) area charts, (C) bar charts, (D) box plots, and (E) up/down charts. Multiple data sets, with multiple values each, are represented per cell as (F) line charts, (G) area charts, (H) bar charts, (I) box plots, and (J) stacked bar charts.

The above examples display only single values, but in many cases more than one data point can be obtained for a given object. A typical such example from proteomics is protein coverage, where both measured sequence coverage as well as (estimated) maximum theoretical coverage for the given experimental conditions can be considered [14–16]. While a typical table would show these values in two separate columns (Fig. 1C), JSparklines can display both values in a single column using stacked bar charts (Fig. 1G). Note that it thus becomes much easier to relate experimental coverage to theoretical coverage for each protein using this approach, while also making it simpler to find the protein with the highest and lowest coverage (and this both in terms of experimental and theoretical coverage).

Another common feature in proteomics is to annotate where peptides are detected in a specific protein sequence. This can of course be shown by listing the position of the aminoterminal residue of the peptides in the given protein as in Fig. 1D. However, this makes it hard to detect peptides that are located close together in the protein, or even to quickly find the most aminoterminal or carboxyterminal peptides identified in the protein sequence. The sparklines visualization, representing the protein sequence by a horizontal line and using colored boxes to indicate the location of the peptides as in Fig. 1H, makes all these tasks straightforward, while adding immediately understandable information about the relative position of the peptides in the protein. Note that this representation also makes it possible to highlight peptides that can map to more than one location on the protein sequence, simply by adding multiple boxes for the peptide in question, as exemplified by Peptide O on the last row. A further refinement is to use colors to indicate the number of times a peptide maps within the protein sequence.

From the above examples, it is clear that sparklines can greatly improve the visual interpretation of data, and that it is particularly effective at increasing the speed and accuracy with which interesting features can be detected. Additional examples of JSparklines can be found in Fig. 2; Fig. 2A–E show visualizations of multiple values per cell, in this case the (simulated) spectrum counting results for 20 peptides for two protein isoforms, while Fig. 2F–J demonstrate the display of multiple data sets, with multiple values each, per cell, here showing the (simulated) spectrum counting results for the same 20 peptides using three different experimental setups (Fig. 2F–J).

In conclusion, the adoption of sparklines has the potential to improve the visual display of proteomic results and scientific data in general. By moving the interpretation from the abstract realm of text and numbers to the more direct and much faster comparison of shapes and colors, complex scientific data can be made more tangible and thus easier to interpret quickly yet accurately.

JSparklines is developed in Java, is open source under the very permissive Apache2 license, and has been tested on Windows, Mac OS X, and Linux. Crossplatform executable binaries, source code, documentation, support, and additional information are available at http://jsparklines.googlecode.com.

H.B. is supported by the Research Council of Norway. L.M. acknowledges the support of Ghent University (Multidisciplinary Research Partnership "Bioinformatics: from nucleotides to networks") and the IWT SBO grant 'InSPECtor' (120025).

The authors have declared no conflict of interest.

References

[1] Tufte, E. R., *The visual display of quantitative information*, Conn. (Box 430, Cheshire 06410): Graphics Press, Cheshire 1983, p 197.

[2] Lee, B., Riche, N. H., Karlson, A. K., Carpendale, S., SparkClouds: visualizing trends in tag clouds. *IEEE Trans. Vis. Comput. Graph* 2010, *16*, 1182–1189.

[3] Radecki, R. P., Medow, M. A., Cognitive debiasing through sparklines in clinical data displays. *AMIA Annu. Symp. Proc.* 2007, *1085*.

[4] Thomas, P., Powsner, S., Data presentation for quality improvement. *AMIA Annu. Symp. Proc.* 2005, *1134*.

[5] Torsvik, T., Lillebo, B., Mikkelsen, G., Presentation of clinical laboratory results: an experimental comparison of four visualization techniques. *J. Am. Med. Inform. Assoc.* 2013, *20*, 325–331.

[6] Bauer, D. T., Guerlain, S., Brown, P. J., The design and evaluation of a graphical display for laboratory data. *J. Am. Med. Inform. Assoc.* 2010, *17*, 416–424.

[7] Wang, R., Fabregat, A., Rios, D., Ovelleiro, D. et al., PRIDE Inspector: a tool to visualize and validate MS proteomics data. *Nat. Biotechnol.* 2012, *30*, 135–137.

[8] Barsnes, H., Martens, L., Crowdsourcing in proteomics: public resources lead to better experiments. *Amino Acids* 2013, *44*, 1129–1137.

[9] Muth, T., Peters, J., Blackburn, J., Rapp, E., Martens, L., ProteoCloud: a full-featured open source proteomics cloud computing pipeline. *J. Proteomics* 2013, *88*, 104–108.

[10] Barsnes, H., Eidhammer, I., Martens, L., FragmentationAnalyzer: An open-source tool to analyze MS/MS fragmentation data. *Proteomics* 2010, *10*, 1087–1090.

[11] Colaert, N., Barsnes, H., Vaudel, M., Helsens, K. et al., Thermo-msf-parser: an open source Java library to parse and visualize Thermo Proteome Discoverer msf files. *J. Proteome Res.* 2011, *10*, 3840–3843.

[12] Vaudel, M., Barsnes, H., Berven, F. S., Sickmann, A., Martens, L., SearchGUI: an open-source graphical user interface for simultaneous OMSSA and X!Tandem searches. *Proteomics* 2011, *11*, 996–999.

[13] Muth, T., Weilnbock, L., Rapp, E., Huber, C. G. et al., DeNovoGUI: an open source graphical user interface for de novo sequencing of tandem mass spectra. *J. Proteome Res.* 2014, *13*, 1143–1146.

[14] Martens, L., Vandekerckhove, J., Gevaert, K., DBToolkit: processing protein databases for peptide-centric proteomics. *Bioinformatics* 2005, *21*, 3584–3585.

[15] Reisinger, F., Martens, L., Database on Demand—an online tool for the custom generation of FASTA-formatted sequence databases. *Proteomics* 2009, *9*, 4421–4424.

[16] Colinge, J., Masselot, A., Carbonell, P., Appel, R. D., InSilicoSpectro: an open-source proteomics library. *J. Proteome Res.* 2006, *5*, 619–624.

