Sparkcals: Small Multiple Calendars for Detecting Temporal Patterns

Ryan Greenberg
School of Information
University of California, Berkeley
ryan@ischool.berkeley.edu

ABSTRACT
Calendar-based visualization sis good for showing patterns, clusters, and gaps predicated on weekly cycles. This paper describes a technique for visualizing time-series data using a small multiple calendar display to facilitate detection of patterns based on weekly schedules. A JavaScript implementation of this technique is presented along with four interactions that can be used to augment this technique.

Author Keywords
visualization, calendars, time-series, small multiples, patterns, trends, interactions

INTRODUCTION
Time-series data is omnipresent in visualization and much work has been done to make it easier to detect patterns over time. Line graphs are useful for presenting long-term phenomena like the change in stock price over years, carbon dioxide emissions over decades, or sunspot cycles over centuries, as well as short-term phenomena like heart rate or electrical activity. The primary variation in these example data is not the result of the daily cycles of human life, however, and line graphs fall short when presenting data that varies based on day of the week.

This is an area where calendars may be useful. Comparatively little work has been done using calendars as an output for data visualization. A calendar is a familiar artifact that can be useful to frame people's understanding of data. Data of human activity in particular is likely to vary based on time of day, and cultural constructions like day of the week and holidays. In these cases, the calendar itself is the cause of variations, making it a potentially valuable tool for presenting them.

Trend and patterns that emerge based on these factors may be less apparent in line graphs at certain scales. We propose a technique to highlight and compare the patterns among different kinds of events on the scale of a month while simultaneously providing a simplified view of events based on time-of-day. This relies on a small multiple display of tiny calendars using the calendar grid as ground, and overlaid information as figure.

This display is augmented by interactions that allow the user to rearrange data as part of the sensemaking process, to zoom to reveal hierarchical relationships, to pivot to switch the relevant dimension being presented, and to merge data to emphasize relationships.

RELATED WORK
A number of techniques have been devised to visualize time-series data in novel ways: using 2D spirals [1], 3D spirals and lattices [2], and bitmap grids [3]. Among these there are some techniques that use calendars directly.

One of the most well known calendar-based visualizations is that of hotel occupancy from Bertin [4]. This work shows the value of multiple calendars to analyze trends across many related variables. Shading on calendars can be used to show variations in data clearly, allowing the viewer to uncover relationships in the dataset. Being able to rearrange

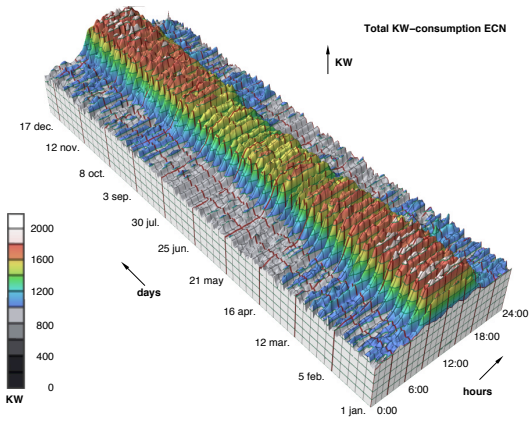

Figure 1. Van Wijk and van Selow present time-series data on a calendar in three dimensions.

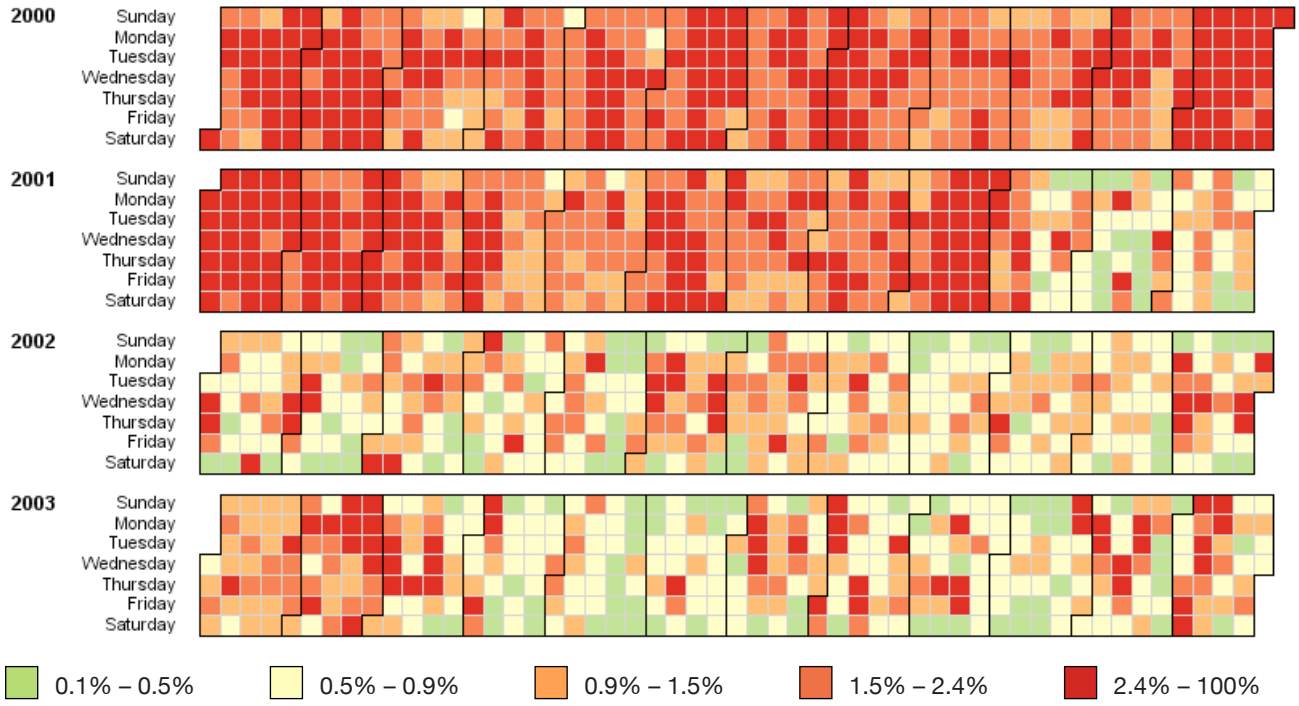

Figure 2. A calendar heatmap showing the percent of U.S. commercial air flights cancelled per day from a visualization by Wicklin and Allison.

the visualization is important because it allows the user to explore different possible relationships among the presented data.

CalendarView is a technique where events are represented by individual pixels on small year-long calendars [5]. This achieves an impressive density, where a 20 pixel square might show 400 events. Ankerst et. al. also discuss the potential benefits of calendar visualizations stemming from preattentive processing of aspects of the calendar, like weekends vs. weekdays.

Van Wijk and van Selow developed a technique designed to present time-series data in a way that shows trends at the level of days, weeks, and years [6]. Their approach uses cluster analysis to identify day patterns of interest and visualizes the result on a calendar. The result is a 3D calendar which shows time of day on the x-axis, days on the y-axis, and a third variable of interest on the z-axis. One limitation of this approach is that details in 3D graphics are difficult to discern due to distortions introduced by foreshadowing.

Calendar heatmaps are an approach that has gone largely unmentioned in the literature [7]. A calendar heatmap is a small depiction of a calendar spanning several months or years where individual days are shaded using different colors that encode some quantitative variable. The layout used by these heatmaps overcomes the aspect ratio problem of a single, extended line chart, but heatmaps suffer from hue's limited limited ability to encode quantitative values.

Finally, design of calendars for visualization can be informed by some visualization principles advocated by Tufte. Proper use of data layering is essential for readable calendars [8]. Although a calendar grid gives

```
{
  "Supermarket":   ["2010/03/29 02:17:43 +0000"],
  "Coffee Shops":  ["2010/04/03 01:31:21 +0000",
                    "2010/04/01 14:56:22 +0000"],
  "UC Berkeley": {
    "Soda Hall":   ["2010/04/21 20:08:37 +0000",
                    "2010/04/19 20:26:47 +0000"],
    "Doe Library": ["2010/04/22 22:12:02 +0000",
                    "2010/04/12 21:20:47 +0000"]
  }
}
```

Figure 3. Data is provided to the sparkcals plugin using the JSON format. Categories are specified as keys which are mapped to arrays of datetime strings or nested objects.

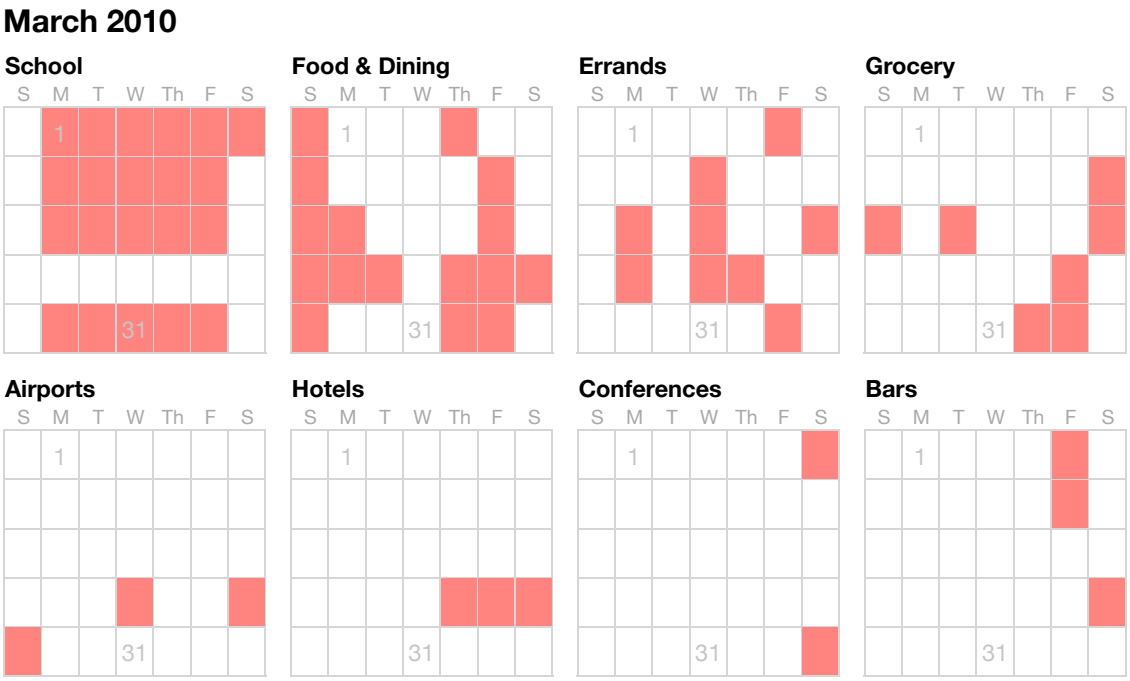

Figure 4. Sparkcals used to visualize an individual's historical location data in eight categories tracked using Foursquare during March 2010. This version uses a single shaded segment to represent all of the time on a given day.

meaning to its content, indicating that the grid is essential to understanding the graphic, it can also be visually distracting if it dominates the presentation. Consequently, the calendar grid should be styled as ground, while the overlaid information is figure. Sparkcals are named after Tufte's work: they aspire to be a calendar version of Tufte's sparklines, which are "data-intense, design-simple, word-sized graphics." cite [9].

METHODS

Our implementation of this technique consists of two plugins written in JavaScript for the popular jQuery library [10]. This architecture makes it simple for end-users to integrate this software into their own web-based projects.

The first plugin generates calendars in HTML with hooks for styling the output. The calendar itself is a `<table>` element. Each day is a `<td>` element divided into horizontal segments specified by the user. Individual cells have CSS classes that correspond to the date (e.g. `2010-05-01`) that permit DOM-based access. When creating calendars the user can provide a function to define the labels for each date, and a function to determine whether a given label is visible upon initialization. These allow the user to minimize the clutter associated with a calendar by labeling, for example, only the first and last day of the month. A sample stylesheet encodes some suggested best-practices for calendar display.

The second plugin accepts data structured as a JavaScript object where keys are mapped to arrays of dates, or other nested objects (see figure 3). The user can also specify a list of time ranges that determine the shaded segments of a given day. By default, each day on the calendar contains a single segment that represents 12:00AM to 11:59PM.

When invoked the sparkcals generates an HTML calendar using the calendar plugin. Then it clones this calendar once for each key at the top-level of the provided object. Where objects are nested they are collapsed recursively to produce a single array of dates. The collection of date associated with each key is transformed to a set of CSS classes that identifies which portions of the current calendar should be shaded. These segments are shaded by finding the corresponding DOM elements and adding a CSS class `shaded`. Each resulting calendar is labeled with its corresponding key, bound to the data used to generate it, and inserted into the browser's DOM at the point specified by the user.

The sparkcals plugin provides an interface to the environment in which it is embedded by using named

Figure 5. An alternate version of the sparkcals display shown in figure 3. This version uses three segments during the day which allows the viewer to interpret the time of events based on the position of the segment within the day.

events for communication. The environment can trigger changes in the interaction mode of the display by calling the *move, zoom, pivot,* or *zoom* methods. The plugin informs the environment of changes in the display by firing events like *modechange* (indicating a successful transition to a new interaction mode), *namechange* (indicating that the environment should change the name of the display), and *details*, providing the environment with the opportunity to provide details-on-demand for portion of the display that are beyond the domain of the plugin.

INTERACTIONS

The current implementation of the sparkcals display includes for four distinct interactions that provide the users with ways to explore the dataset.

In *move* mode, the user can rearrange the calendars in the display to group ones that appear to be related, or to place two calendars the user wants to compare next to each other. The ability to re-arrange calendars addresses a common issue with small multiple displays, where it can be difficult to compare instances that are located at a distance. This mode is based on Bertin's idea that arranging data is an important part of sensemaking.

Second, the user can *zoom* to reveal the underlying hierarchy of data. At the top level, events may have been collapsed into the presented categories. By zooming into a category, the user can see what subcategories or elements comprise the category and lead to the patterns presented. If the user is curious

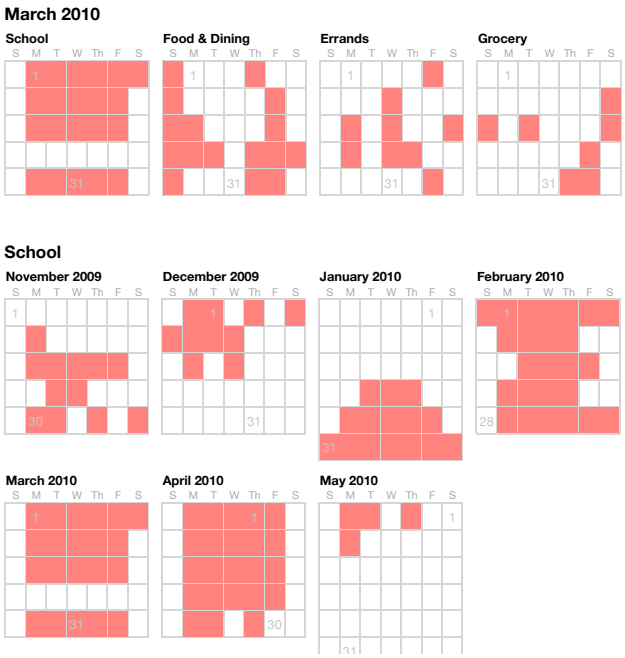

Figure 6. The pivot tool lets the user change whether a given category is compared with others at the same time or with itself over time. The lower display shows the school category after a pivot around school.

about the origin of a certain pattern shown for a high level category zooming allows for exploration of this.

Third, the *pivot* tool allows the user to change the dimension being displayed. By default, the sparkcal display shows the same month once for each category at the current level of the dataset. If the user is interested in a particular category and how it has

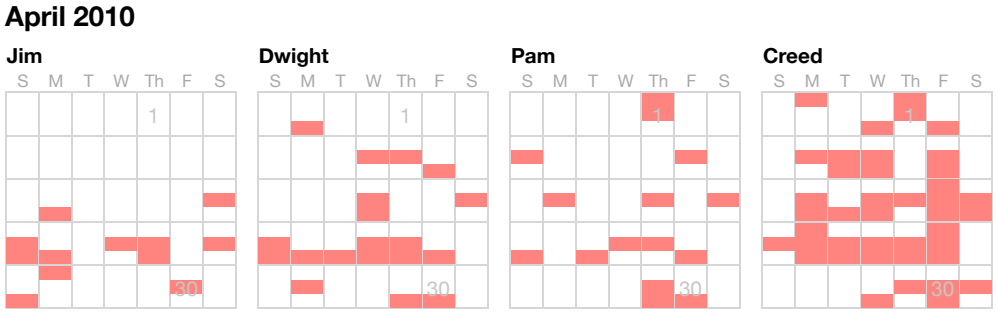

Figure 7. A sparkcal visualization of the commit history in a Subversion repository. Based on the position of the shaded segments, we can see the relative infrequency of commits made before 12:00PM.

changed over time, she can pivot on that calendar to show that category over time instead of that category compared with others.

Finally, the user can *merge* two calendars to show their events on the same grid.

RESULTS

We tested sparkcal technique using the checkin history of an individual using Foursquare, the location-based social network. When a Foursquare user goes to a venue or point of interest, she reports her location and selecting from a list of known establishments at that place. The result can be transformed into a list of venues and the date and times of visits. Using user-defined categories or those provided by Foursquare itself, we can group venues into appropriate categories.

Figures 4 and 5 shows the resulting sparkcals visualization of a user's location history during the month of March 2010. Consider the first calendar in the display, which shows the user's visits to school in March. A quick observation reveals two dominant patterns: the presence of shading during the weekdays and the absence of shading during the weekends. It also shows two aberrations from these patterns: there is a single Saturday that is shaded, and the fourth week of the month is empty. The other calendars in the display offer insight into possible explanations. The lone Saturday shaded as school is also shaded as a conference. During the week of absence from school there are shaded cells at airports and hotels indicating travel. This week was spring break at UC Berkeley, and although this data is not available in the display itself, the display gives clues that prompt the viewer to ask questions about it.

A day can have an arbitrary number of shaded segments, although a small number will probably be the most intelligible. Figure m shows the same result as figure n, except that days have been divided into segments that represent midnight to noon (morning), noon to 6:00PM (afternoon), and 6:00PM to midnight (evening). In this example viewers can use the position of the segment—whether it is flush against the top or bottom, or aligned in the middle—to detect trends that occur based on time of day.

When the user zooms into a a category the current small multiple display is replaced with a new display of the subcategories that comprised the upper-level. This lets users view nested sets of data in an interactive fashion.

When the user pivots around a calendar, the display shifts from showing all the categories contained within a given month to showing the category that was the focus of the pivot across several months.

Since the current implementation of this system is written in JavaScript, its efficiency depends greatly on the optimization of the runtime in which it is executed. On an Intel 2.2 Ghz Core 2 Duo processor, initializing and drawing a sparkcals display with 10 calendars and 70 shaded segments based on 373 dates takes 240ms in Google Chrome, 400ms in Safari 4, and 775ms in Firefox 3.6. The current implementation is particularly inefficient because it relies on DOM traversal and manipulation to shade calendar segments instead of creating the end result entirely in HTML and having the browser parse it in a single pass. A non-browser-based implementation could be orders of magnitude faster.

DISCUSSION

Since line graphs and bar charts are quite good at showing univariate change over time, any competing technique must offer some benefit not afforded by these methods. A calendar can be thought of as a kind

of bar chart, with seven-day slices arranged on top of one another instead of along a single x-axis. This arrangement makes some comparisons easier. A viewer can look up and down to see, for example, how all the Fridays compare. By overlaying data on a calender grid sparkcals take advantage of this arrangement and activate people's familiarity with the shape of calendars. This may promote faster recognition of patterns that occur based on calendar cycles.

Small multiple displays are one useful way to answer the question, "compared to what?" In its default instantiation, sparkcals allow the viewer to see categorical data for a one-month timespan and make comparisons. By using the pivot tools, the viewer can switch from comparing categories to examining change in a single category over an extended period. This works nicely in an interactive display, but is not effective for static graphics.

There are some potential drawbacks to using sparkcals. In some cases, for a particularly large number of categories, they may be too sparse to be an effective way to visualize the data. If all the events were displayed on a single calendar, perhaps using hue to encode categories, the result would be a much more dense display, although the changes in trends might not be as apparent. Sparkcals are also unlikely to be a useful approach for data without major trends predicated on the day-of-the week or time of day. When trends are expected on the order of months, or the intent is to view changes over a long period of time, existing visualization techniques may be better suited to the intent.

FUTURE WORK

Our work has been an initial attempt at developing this new technique. Future work could focus on testing the effectiveness of sparkcals empirically to see if they enhance viewer's ability or speed in detecting patterns. Additional features could include display of multiple months of calendar data, like blocks of three months; using the brightness of a shaded segment to encode the frequency of an event occurring within that band; or more intelligent association of shaded segments with corresponding days. People often think of the earliest hours of the day as an extension of the previous day, so depending on the task at hand showing midnight – 2:00AM, e.g., as part of the preceding day might give a better impression of what happened. For the sake of internationalization, it would be helpful if Monday could be positioned as the first day of the week.

REFERENCES

1. Weber, M., Alexa M., and Müller W., Visualizing time-series on spirals. *IEEE Symposium on Information Visualization*, 2001, p. 7.
2. Mackinlay, J.D., Robertson, G.G. and DeLine, R., Developing calendar visualizers for the information visualizer. *7th annual ACM symposium on User interface software and technology*, 1994, p. 118.
3. Kumar, N., Lolla, N., Keogh, E.., Lonardi, S., Ratanamahatana, C., and Wei L.. Time-series bitmaps: a practical visualization tool for working with large time series databases. SIAM 2005 Data Mining Conference, 2005, pp. 531–535.
4. Bertin, J., *Semiology of graphics*, University of Wisconsin Press, 1983.
5. Ankerst, M., Jones, D.H., Kao, A. and Wang, C. DataJewel: Tightly integrating visualization with temporal data mining. *ICDM Workshop on Visual Data Mining*, 2003.
6. Van Wijk, J.J. and Van Selow, E.R., Cluster and calendar based visualization of time series data. infovis, 1999, p. 4.
7. Wicklin, R., and Allison, R. Congestion in the sky: Visualising domestic airline traffic with SAS. *ASA Data Expo 2009*. Retrieved from http://stat-computing.org/dataexpo/2009/posters/wicklin-allison.pdf
8. Tufte, E.R. The visual display of quantitative information, Cheshire, CT: Graphics Press, 2001.
9. Tufte, E.R. *Envisioning Information*, Cheshire, CT: Graphics Press, 1990.
10. jQuery. http://www.jquery.com.