Viewing Data: The rdataviewer Package

Paul Murrell
The University of Auckland

Abstract

The rdataviewer package is a prototype software tool for the R environment that implements several new ideas for viewing data sets. The “shape” of the entire data set is always visible to the user, to assist with learning about data structures. In addition, the view of the data set may be interactively zoomed and, for external text files, only the visible part of the data set is loaded into memory. Together these allow very large data sets to be viewed and navigated.

Keywords: R, data structures, large data.

1. Introduction

This article describes several new ideas for ways to view data files and data structures, with applications to teaching and viewing large data sets. These ideas are implemented in a prototype package for the R system (R Development Core Team 2011) called rdataviewer (Murrell 2011).

1.1. Teaching students about data structures

One of the difficulties that many students face when they first encounter the R environment, particularly those students without a background in computing, is the concept of data structures. Something that does not help the students’ plight is the way that data structures are displayed on screen. For example, in R, a simple vector of values is conceptually a 1-dimensional data structure, typically thought of as a column of values. However, due to the limitations of screen real estate, a vector is printed on screen horizontally, with values wrapping across several lines when there are too many values to be displayed on one line. The following simple example, consisting of a character vector containing the letters of the english alphabet, suffices to demonstrate the problem. This does not look like very much like a single column of values to the uninitiated.
A conceptually 2-dimensional structure, like a matrix, can actually be easier to grasp in simple cases because the values are displayed by R in rows and columns, as shown below.

```r
R> head(VADeaths)

Rural Male Rural Female Urban Male Urban Female
50-54 11.7  8.7  15.4  8.4
55-59 18.1 11.7  24.3 13.6
60-64 26.9 20.3  37.0 19.3
65-69 41.0 30.9  54.6 35.1
70-74 66.0 54.3  71.1 50.0
```

However, for anything but very small data sets, this convenient conceptual and visual correspondence rapidly breaks down. Again, the uninitiated can struggle to see that the following output represents a 2-dimensional structure with 6 rows and 11 columns. More columns just make the problem worse.

```r
R> head(mtcars)

mpg cyl disp hp drat wt qsec vs am
Mazda RX4  21.0  6 160 110 3.90 2.620 16.46 0 1
Mazda RX4 Wag 21.0  6 160 110 3.90 2.875 17.02 0 1
Datsun 710  22.8  4 108 93 3.85 2.320 18.61 1 1
Hornet 4 Drive 21.4  6 258 110 3.08 3.215 19.44 1 0
Hornet Sportabout 18.7  8 360 175 3.15 3.440 17.02 0 0
Valiant  18.1  6 225 105 2.76 3.460 20.22 1 0
gear carb
Mazda RX4  4  4
Mazda RX4 Wag 4  4
Datsun 710  4  1
Hornet 4 Drive 3  1
Hornet Sportabout 3 2
Valiant  3  1
```

One solution is to arrange the display of values so that there is a correspondence with the conceptual shape of the data structure. For example, in a spreadsheet view, as shown by the `View()` function in R, or the more sophisticated data viewers provided by GUIs such as `Remdr` (Fox 2011) or `JGR` (Helbig, Urbanek, and Fellows 2010), the values of a vector appear in a single column and the values in a data frame are shown in rows and columns (see Figure 1). However, because the entire data structure is not visible, the overall shape is still unknown. Students easily lapse into thinking that the shape corresponds only to those values that are visible.
Figure 1: A vector (left) and a data frame (right) as shown by the `View()` function in R.

A solution advocated in this paper is to supplement the spreadsheet view with an image that presents the overall shape of the entire data structure when viewing values from the data structure. For example, Figure 2 shows two consoles that display the shapes of the full `letters` vector and the full `mtcars` data frame.

1.2. Viewing large data sets

Printing out the raw data values in a data structure, as shown in the previous section, may seem like a somewhat pointless exercise. It is rarely the case, except for very small data sets, that we actually need to view an individual data value within a data set because interest more often focuses on the results of analyses or summaries of the raw data values. Even for the purpose of checking data, it makes more sense to view data summaries or plots rather than attempting to view thousands of individual raw data values.

Nevertheless, it is often useful to view at least a sample of the values within a data set, if only to get an impression of the sorts of data values that lie within. For example, although we might not want to view all 578 rows of the `ChickWeight` data set, it is still useful to view the first few rows to gain an impression of the data values within each of the four columns, as shown below.

```r
R> head(ChickWeight)
weight Time Chick Diet
1 42 0 1 1
2 51 2 1 1
3 59 4 1 1
4 64 6 1 1
```
Viewing Data

Figure 2: Two examples of simple images (the rectangular areas above the two buttons) that show the overall shape of a vector (left) and a data frame (right). These should be compared with the restricted view of these data that is typical of existing data viewers (see Figure 1).

If a data set is stored within a text file, it is also useful to view the raw values within the text file in order to discover the text format that has been used.

With appropriate subsetting tools, such as those available in R, it is also relatively straightforward to view any simple subset of a data set (not just the first few rows). For example, the following code views the 115th to the 120th rows of the ChickWeight data set.

R> ChickWeight[115:120,]

However, this is an example of a simple task that can be performed much more conveniently via a graphical user interface, where “Page Up” and “Page Down” keys, or a scroll bar, allow for much faster exploration of multiple views of the data. Again, a spreadsheet type of view, as provided by the View() function in R, provides this sort of facility.
The problem comes when the data set that we want to browse is large. In such cases, it can still be useful to view raw values (as we will see in Section 3.2), but standard spreadsheet or even text viewing software may not be able to cope.

Two problems in particular are addressed in this article: being able to view a large data set or a large text file at all; and being able to view a useful portion of the data set or text file. In the former case, viewing software may struggle to even open a file that is hundreds of megabytes in size, or larger. In the latter case, if a data set has hundreds of columns and thousands of rows, it is useful to be able to decide how many of those rows and columns are currently viewed. Only being able to view 10 columns and 50 rows at a time is less useful than also being able to view 100 columns and 5000 rows at a time if we so choose.

The proposed solution to the first problem is to provide a viewer tool that only loads into computer memory as much of the data set as is required to show the current view of the data. This allows data sets of arbitrary size to be viewed. The proposed solution to the second problem is to provide a viewer tool that allows interactive zooming of the current view, to allow more rows and columns to be viewed at once. For example, Figure 3 shows two different views of the same data set at different levels of zoom.

2. The rdataviewer package

The rdataviewer package is a software prototype for trying out the ideas described in Section 1. The main function in the package is called view() and it takes a single argument, which is the data set to view. For example, the following code is used to view the mtcars data frame.

```
R> library("rdataviewer")
R> view(mtcars)
```

The R graphics window displays the current view of the data set. This will typically be just a portion of the data set, as in Figure 4. The Tcl/Tk window serves two functions: it allows the user to modify the current view of the data set (through various key strokes) and it displays a diagram of the overall shape of the data set with a red rectangle to represent what subset of the data is currently being viewed. In Figure 4, we see that the mtcars data set is a square-ish two-dimensional structure (grey and white regions are used to indicate different columns within the data set) and we are currently viewing roughly the top-left quarter of the data.

Navigation of the current view is row- and column-based. For example, the right arrow navigates the view one column to the right and the down arrow navigates the current view one row down. The page down and page up keys, plus control-home and control-end also work. The Tcl/Tk window must be the active window, otherwise key strokes are ignored.

Of more relevance to this article are the key strokes for zooming the current view. With the shift key down, the right arrow increases the zoom factor so that at least one fewer column is visible. With the control key down, the right arrow decreases the zoom factor so that at least one more column is visible. For example, Figure 5 shows the two rdataviewer windows
Figure 3: Two views of a data set with 148 rows and 13 columns. On top is the default view and below that the view has been zoomed so that all rows of the data are visible. When text becomes very small, as in the bottom window, there is an option to draw simple lines of the appropriate length rather than text.
after the right arrow key has been pressed twice, with the control key held down; now all of the wt and qsec columns are also visible. Notice also that the red region within the Tcl/Tk window has grown to reflect the fact that a larger portion of the data is now visible.

This simple example demonstrates the first idea of showing the overall shape of the data set while viewing only a subset of the actual values, plus the idea of allowing simple zooming to control how much of the data set is being viewed. The next section looks at part of the design of the rdataviewer package in more detail in order to demonstrate the ideas for viewing large data sets.

3. The design of rdataviewer

The rdataviewer package is based on four fundamental classes: a ViewerData object provides the data set to be viewed; a ViewerState object records the current view of the data (what portion of the data is visible); a ViewerDevice object provides somewhere to draw the current view of the data; and a Viewer object contains one of each of the three other objects, provides an interface to change the current view, and coordinates the updating of the ViewerDevice with changes in the ViewerState.

The rdataviewer package provides a Viewer interface that is based on the tcltk GUI toolkit. This allows the package to be used without having to install a separate GUI toolkit like GTK+ or Qt. However, the design allows for different front ends to be added. The package also provides a ViewerDevice based on the current R graphics device and a default ViewerState. All we need to provide for the view() function is a ViewerData object. The view() function will accept raw R data structures as we saw in the previous section, but we can obtain greater control if we explicitly generate a ViewerData object ourselves. For example, the result shown
8 Viewing Data

Figure 5: The two windows from after two right arrow key strokes, with the control key held down. The current view has zoomed out so that two more complete columns are now visible.

in Figure 4 could also have been obtained with the following code, which explicitly creates a ViewerData object from the mtcars data frame.

R> view(viewerDataFrame(mtcars))

3.1. The ViewerData class

A ViewerData object contains the actual data set that is to be viewed and it is required to provide the following information about the data set:

dimensions: The total number of rows and columns in the data set.
column widths: The width of each column (number of characters).
column names: A name for each column (for a specified range of columns).
text representation: A text version of the data set (for a specified range of rows and columns).

For example, when the data set is a data frame, like mtcars, the viewerDataFrame() function creates a ViewerData object containing the data frame and it calculates the dimensions of the data set using the dim() function. The column names, column widths, and text representation of the data set are all obtained by capturing the output of a call to print() for the appropriate subset of the data frame (the rows and columns of the data frame that are currently visible).

The reason for this design of the ViewerData class is that it allows new classes to be derived for other kinds of data sets.
3.2. Extending the ViewerData class

The viewerDataFrame() function creates an object of class ViewerDataFrame, which extends the ViewerData class and allows R data frames to be viewed with the view() function. We can extend the ViewerData class in other ways to allow other data sources to be viewed. For example, the viewerDataText() function creates an object of class ViewerDataText, which allows plain text files to be viewed.

A ViewerDataText object contains the name of a text file and provides the required information from the file in the following ways:

- **dimensions**: There is only one “column” and, when the ViewerDataText object is created, the file is searched for line breaks to determine how many lines (“rows”) it has.
- **column widths**: The width of the single column is calculated from the maximum number of characters between line breaks.
- **column names**: This is just the name of the file.
- **text representation**: A file “seek” is used to read only the rows of the file that are currently being viewed.

This implementation allows very large files to be viewed because only the viewed portion of the file is ever read into memory. For example, the following code shows a view of the start of a 300 MB XML file with 4,772,013 lines (see Figure 6).

```R
R> view(viewerDataText("/scratch/Metrix/BlindData/MEENDL30102008_001-blind.XML", R+ index=TRUE))
```

Although it may take a few seconds for the windows to appear, just being able to view this file is a minor victory. For example, heavyweight text editors like vi (Moolenaar 2006) and Emacs (Stallman 2002) struggle with it (vi takes several minutes to open the file and Emacs fails to open it at all).

Furthermore, navigation within the file is almost instantaneous. For example, it is possible to navigate to the middle of the file by typing 2000000 and then G. It is also possible to zoom the view as normal. For example, Figure 7 shows the view after pressing Page Up several times, with the control key held down. In this view, the raw data values are not legible, but it is still interesting to see and explore the regular structure of the raw values. Also shown in Figure 7 is this zoomed view after browsing down the file by pressing Page Down several times. This shows a clear change in the data.

We can zoom in to this area of the file to see that the change involves the disappearance of QUALITYFLAG values after row 2936 (see Figure 8).

The ViewerData class has also been extended via the ViewerDataMySQL class so that an external MySQL database can be used as a data source for the view() function (see the viewerDataMySQL() function).

4. Discussion

The rdataviewer package is a software prototype for experimenting with several new ideas for viewing data sets. The package has one main function, view(), which creates two windows:
Figure 6: The rdataviewer windows for viewing the Metrix data set (a 300 MB XML file).

Figure 7: Two different views of the Metrix data set. On the left, the original view (Figure 6) has been zoomed out so that only the regular structure of the data values is visible. On the right, this zoomed view has been browsed to discover a phase change in the structure of the data values.
Figure 8: A zoomed view of the phase change in the structure of the Metrix data set (see Figure 7).
one for viewing the data set and one for controlling what portion of the data set is visible. A feature of the latter window is a diagrammatic representation of the overall shape of the data set being viewed, which may help students to understand the nature of data structures. A feature of the navigation interface is that it is possible to zoom the view of the data, so that useful amounts of large data sets can be viewed at once. The rdataviewer package is designed so that only the viewed portion of the data set is needed in memory at any one time. This allows large data sets to be viewed and browsed.

The package is built upon a set of S4 classes and generic functions so that it is easy for others to experiment further with alternative user interfaces, alternative displays of the data, and alternative data sources.

4.1. Future directions

The rdataviewer package is primarily a test implementation for trying out different ideas for viewing data sets. It is hoped that some of these ideas might be adopted in more complete, sophisticated, and user-friendly systems, not necessarily restricted to R.

Nevertheless, the package itself may provide a useful tool if no other tool exists for navigating large raw data sets. Furthermore, the package may provide a basis for further experimentation, particularly with regard to alternative data sources. For example, it would be interesting to experiment with an interface to data structures that are stored on disk via the ff package (Adler, Gläser, Nenadic, Oehlschlägel, and Zucchini 2011).

The demonstration in this article is limited to data set formats with a rectangular structure (vectors, data frames, and plain text files). There is plenty of scope to explore other sorts of data structures, particularly recursive structures like lists.

The implementation considered in this article also only considers the task of viewing the data. It would also be useful to experiment with allowing editing of data values, if only to transform the data (for example, reordering of the rows).

References


**Affiliation:**

Paul Murrell  
Department of Statistics  
The University of Auckland  
38 Princes Street, Auckland  /New Zealand  
E-mail: paul@stat.auckland.ac.nz  
URL: http://www.stat.auckland.ac.nz/~paul/