



Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II.

<http://www.jstatsoft.org/>

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.

```
R> letters
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

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> 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> 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 **Rcmdr** (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.

	row.names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
2	Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
6	Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
7	Duster 360	14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.44	18.3	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.44	18.9	1	0	4	4
12	Merc 450SE	16.4	8	275.8	180	3.07	4.07	17.4	0	0	3	3
13	Merc 450SL	17.3	8	275.8	180	3.07	3.73	17.6	0	0	3	3
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.78	18	0	0	3	3
15	Cadillac Fleetwood	10.4	8	472	205	2.93	5.25	17.98	0	0	3	4
16	Lincoln Continental	10.4	8	460	215	3	5.424	17.82	0	0	3	4
17	Chrysler Imperial	14.7	8	440	230	3.23	5.345	17.42	0	0	3	4
18	Fiat 128	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4	1
19	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
20	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4	1
21	Toyota Corona	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3	1
22	Dodge Challenger	15.5	8	318	150	2.76	3.52	16.87	0	0	3	2
23	AMC Javelin	15.2	8	304	150	3.15	3.435	17.3	0	0	3	2
24	Camaro Z28	13.3	8	350	245	3.73	3.84	15.41	0	0	3	4
25	Pontiac Firebird	19.2	8	400	175	3.08	3.845	17.05	0	0	3	2

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> 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

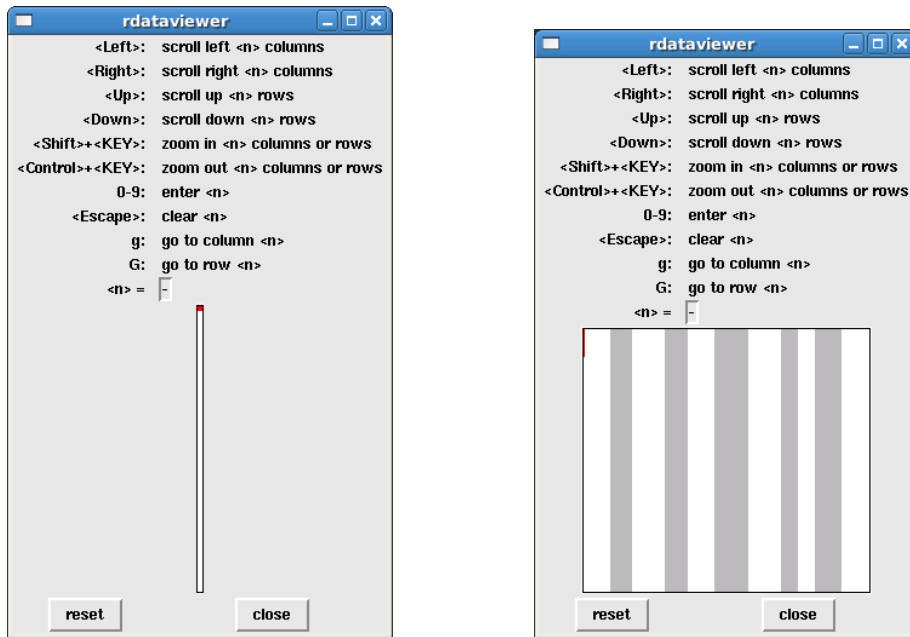


Figure 2: Two examples of simple images (the rectangular areas above the two buttons) that show the overall shape of a vector (left) and a data frame (right). These should be compared with the restricted view of these data that is typical of existing data viewers (see Figure 1).

```
5    76    8    1    1
6    93   10    1    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, ]
```

```
      weight Time Chick Diet
115      96   14    10    1
116     101   16    10    1
117     112   18    10    1
118     120   20    10    1
119     124   21    10    1
120      43    0    11    1
```

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. The result is a pair of windows, one an R graphics window and one a Tcl/Tk window (see Figure 4).

```
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

The top screenshot shows the following data:

	Planet	Star
1	PER 1257 a	PER 1257
2	PER 1257 b	PER 1257
3	PER 1257 c	PER 1257
4	PER 1257 d	PER 1257
5	51 Pegasi b	51 Pegasi
6	Upsilon Andromedae b	Upsilon Andromedae
7	55 Cancri b	55 Cancri
8	47 Ursae Majoris b	47 Ursae Majoris
9	tau Boo	tau Bootis
10	70 Virginis b	70 Virginis
11	rho CrB	rho Coronae Borealis
12	16 Cygni b	16 Cygni
13	HD 217107 b	HD 217107
14	HD 210277 b	HD 210277
15	HD 187123 b	HD 187123
16	Gliese 876 b	Gliese 876
17	HD 195019	HD 195019
18	HD 168443 b	HD 168443
19	HD 168443 c	HD 168443
20	14 Herculis b	14 Herculis
21	HD 209458 b	HD 209458
22	HD 192263 b	HD 192263
23	HD 37124 b	HD 37124
24	HD 130322 b	HD 130322
25	HD 177830 b	HD 177830
26	HD 134987 b	HD 134987
27	HR 810 b	HR 810
28	Upsilon Andromedae c	Upsilon Andromedae
29	Upsilon Andromedae d	Upsilon Andromedae
30	HD 222582 b	HD 222582
31	HD 10697 b	HD 10697
32	HD 83443 b	HD 83443
33	HD 168746 b	HD 168746
34	HD 46375 b	HD 46375
35	HD 108147 b	HD 108147

The bottom screenshot shows the same data set zoomed to fit all 148 rows, with text rendered as horizontal lines.

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.

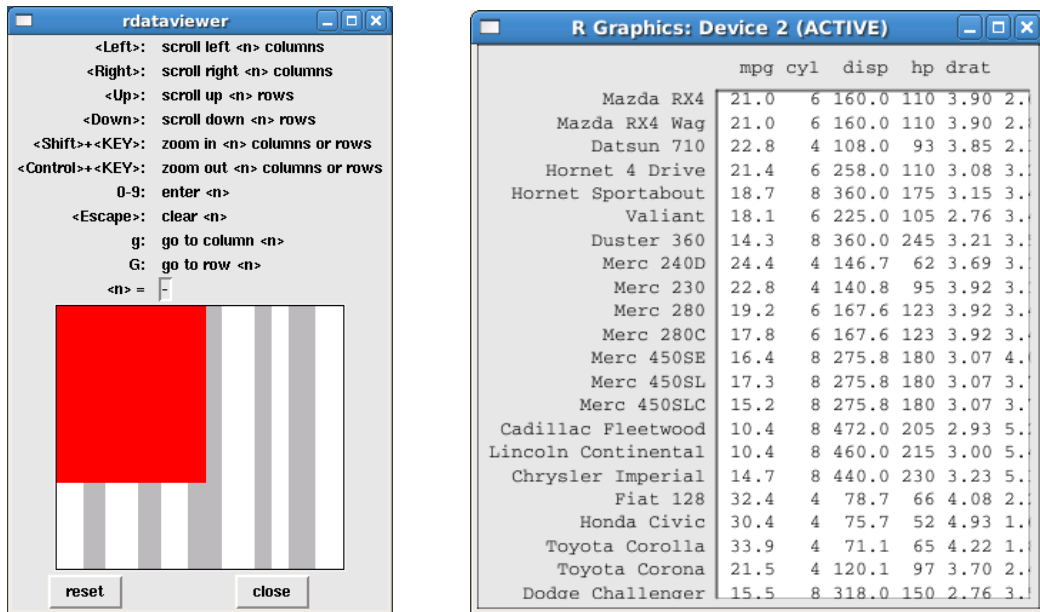


Figure 4: The two windows that are generated by the `view()` function: a Tcl/Tk window (on the left) and an R graphics window (on the right).

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

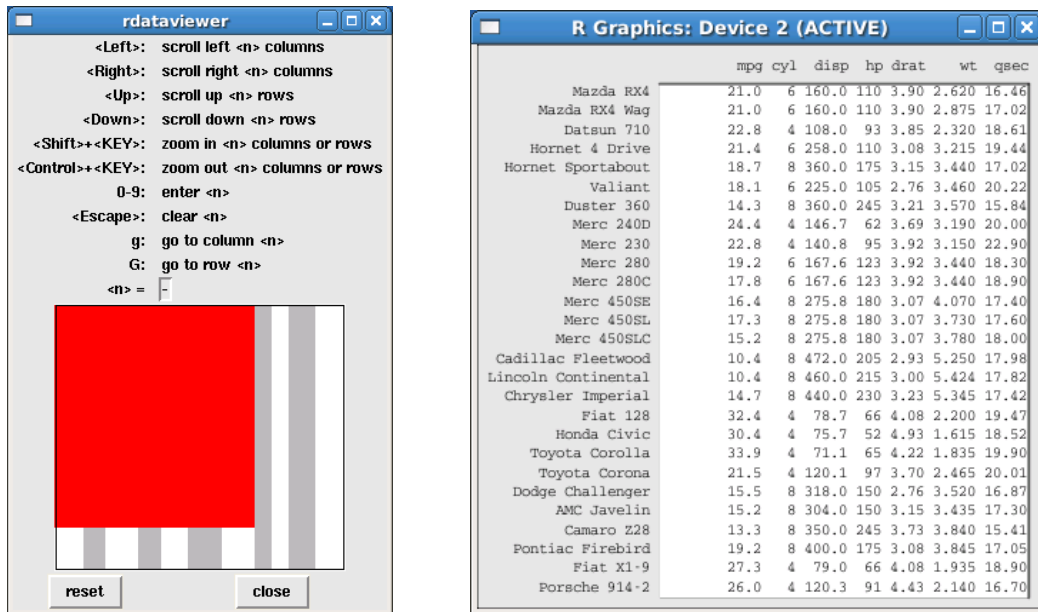


Figure 5: The two windows from 4 after two right arrow key strokes, with the control key held down. The current view has zoomed out so that two more complete columns are now visible.

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> 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 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:

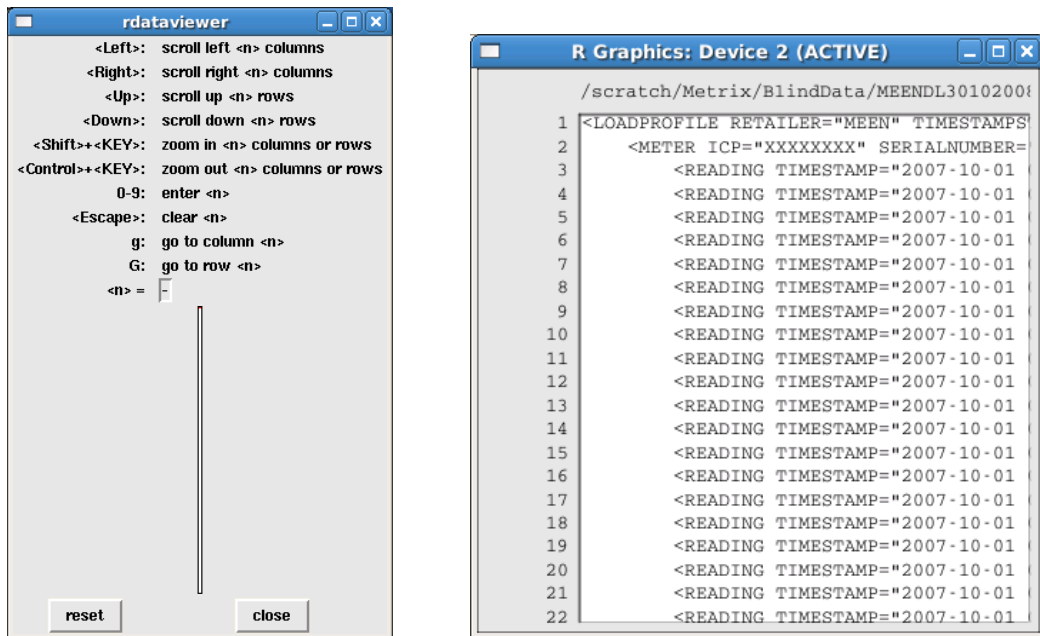


Figure 6: The **rdataviewer** windows for viewing the Metrix data set (a 300 MB XML file).

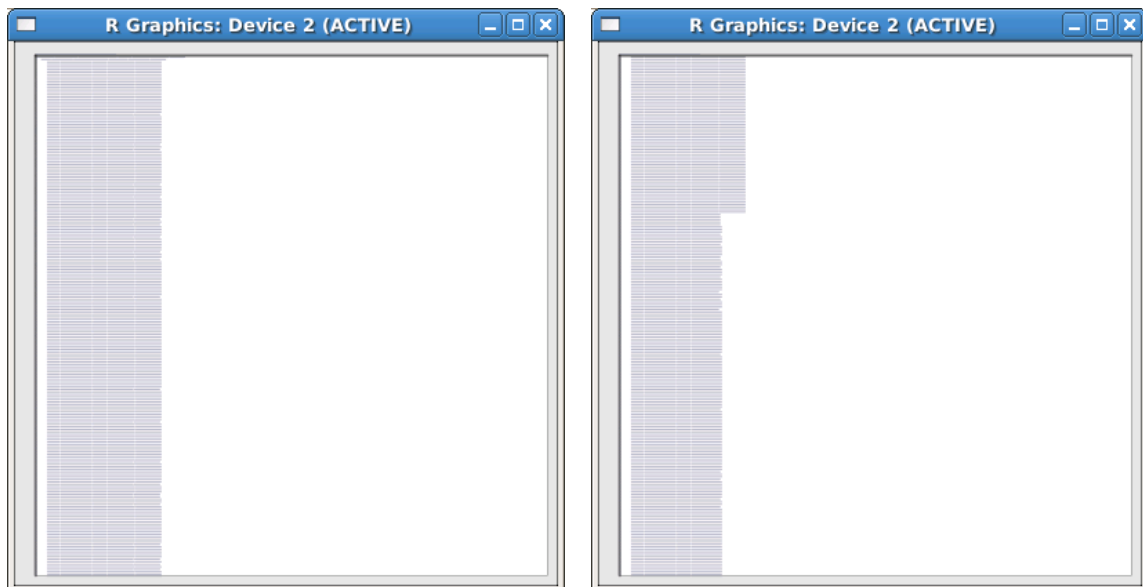
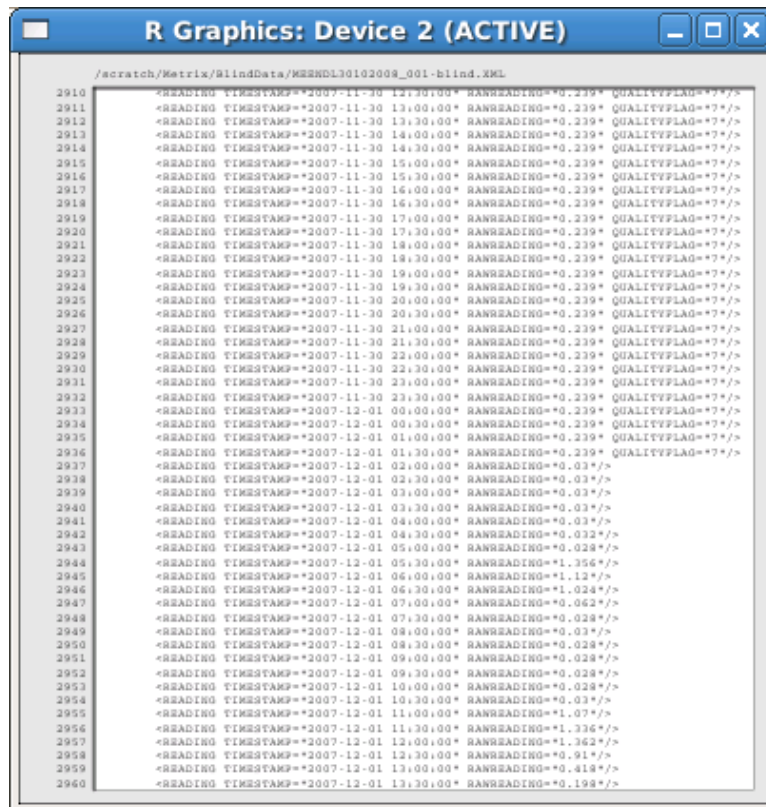


Figure 7: Two different views of the Metrix data set. On the left, the original view (Figure 6) has been zoomed out so that only the regular structure of the data values is visible. On the right, this zoomed view has been browsed to discover a phase change in the structure of the data values.



```

R Graphics: Device 2 (ACTIVE)
/scratch/Matrix/BlindData/MEENHL30102008_001-blind.XML
2910 <READING TIMESTAMP="2007-11-30 12:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2911 <READING TIMESTAMP="2007-11-30 13:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2912 <READING TIMESTAMP="2007-11-30 13:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2913 <READING TIMESTAMP="2007-11-30 14:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2914 <READING TIMESTAMP="2007-11-30 14:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2915 <READING TIMESTAMP="2007-11-30 15:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2916 <READING TIMESTAMP="2007-11-30 15:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2917 <READING TIMESTAMP="2007-11-30 16:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2918 <READING TIMESTAMP="2007-11-30 16:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2919 <READING TIMESTAMP="2007-11-30 17:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2920 <READING TIMESTAMP="2007-11-30 17:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2921 <READING TIMESTAMP="2007-11-30 18:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2922 <READING TIMESTAMP="2007-11-30 18:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2923 <READING TIMESTAMP="2007-11-30 19:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2924 <READING TIMESTAMP="2007-11-30 19:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2925 <READING TIMESTAMP="2007-11-30 20:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2926 <READING TIMESTAMP="2007-11-30 20:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2927 <READING TIMESTAMP="2007-11-30 21:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2928 <READING TIMESTAMP="2007-11-30 21:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2929 <READING TIMESTAMP="2007-11-30 22:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2930 <READING TIMESTAMP="2007-11-30 23:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2931 <READING TIMESTAMP="2007-11-30 23:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2932 <READING TIMESTAMP="2007-11-30 23:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2933 <READING TIMESTAMP="2007-12-01 00:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2934 <READING TIMESTAMP="2007-12-01 00:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2935 <READING TIMESTAMP="2007-12-01 01:00:00" SANREADING="0.239" QUALITYFLAG="*/>
2936 <READING TIMESTAMP="2007-12-01 01:30:00" SANREADING="0.239" QUALITYFLAG="*/>
2937 <READING TIMESTAMP="2007-12-01 02:00:00" SANREADING="0.03"/>
2938 <READING TIMESTAMP="2007-12-01 02:30:00" SANREADING="0.03"/>
2939 <READING TIMESTAMP="2007-12-01 03:00:00" SANREADING="0.03"/>
2940 <READING TIMESTAMP="2007-12-01 03:30:00" SANREADING="0.03"/>
2941 <READING TIMESTAMP="2007-12-01 04:00:00" SANREADING="0.03"/>
2942 <READING TIMESTAMP="2007-12-01 04:30:00" SANREADING="0.032"/>
2943 <READING TIMESTAMP="2007-12-01 05:00:00" SANREADING="0.028"/>
2944 <READING TIMESTAMP="2007-12-01 05:30:00" SANREADING="1.256"/>
2945 <READING TIMESTAMP="2007-12-01 06:00:00" SANREADING="1.12"/>
2946 <READING TIMESTAMP="2007-12-01 06:30:00" SANREADING="1.024"/>
2947 <READING TIMESTAMP="2007-12-01 07:00:00" SANREADING="0.062"/>
2948 <READING TIMESTAMP="2007-12-01 07:30:00" SANREADING="0.028"/>
2949 <READING TIMESTAMP="2007-12-01 08:00:00" SANREADING="0.03"/>
2950 <READING TIMESTAMP="2007-12-01 08:30:00" SANREADING="0.028"/>
2951 <READING TIMESTAMP="2007-12-01 09:00:00" SANREADING="0.028"/>
2952 <READING TIMESTAMP="2007-12-01 09:30:00" SANREADING="0.028"/>
2953 <READING TIMESTAMP="2007-12-01 10:00:00" SANREADING="0.028"/>
2954 <READING TIMESTAMP="2007-12-01 10:30:00" SANREADING="0.03"/>
2955 <READING TIMESTAMP="2007-12-01 11:00:00" SANREADING="1.07"/>
2956 <READING TIMESTAMP="2007-12-01 11:30:00" SANREADING="1.236"/>
2957 <READING TIMESTAMP="2007-12-01 12:00:00" SANREADING="1.262"/>
2958 <READING TIMESTAMP="2007-12-01 12:30:00" SANREADING="0.91"/>
2959 <READING TIMESTAMP="2007-12-01 13:00:00" SANREADING="0.418"/>
2960 <READING TIMESTAMP="2007-12-01 13:30:00" SANREADING="0.198"/>

```

Figure 8: A zoomed view of the phase change in the structure of the Matrix 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

- Adler D, Gläser C, Nenadic O, Oehlschlägel J, Zucchini W (2011). *ff: Memory-efficient storage of large data on disk and fast access functions*. R package version 2.2-3, URL <http://CRAN.R-project.org/package=ff>.
- Fox J (2011). *Rcmdr: R Commander*. R package version 1.6-3, URL <http://CRAN.R-project.org/package=Rcmdr>.
- Helbig M, Urbanek S, Fellows I (2010). *JGR: Java Gui for R*. R package version 1.7-1, URL <http://CRAN.R-project.org/package=JGR>.
- Moolenaar B (2006). *vim - Vi IMproved*. Version 7.0, URL <http://www.vim.org/>.
- Murrell P (2011). *rdataviewer: Data Viewer*. R package version 0.1, URL <http://www.stat.auckland.ac.nz/~paul/>.
- R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org/>.

Stallman R (2002). *GNU Emacs*. Version 21.4.1, URL <http://www.gnu.org/s/emacs/>.

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/>