Journal of Statistical Software

March 2014, Volume 57, Issue 5. *http://www.jstatsoft.org/*

Design of Diverging Stacked Bar Charts for Likert Scales and Other Applications

Richard M. Heiberger
Temple University

Naomi B. Robbins
NBR

Abstract

Rating scales, such as Likert scales, are very common in marketing research, customer satisfaction studies, psychometrics, opinion surveys, population studies, and numerous other fields. We recommend diverging stacked bar charts as the primary graphical display technique for Likert and related scales. We also show other applications where diverging stacked bar charts are useful. Many examples of plots of Likert scales are given. We discuss the perceptual and programming issues in constructing these graphs. We present two implementations for diverging stacked bar charts. Most examples in this paper were drawn with the `likert` function included in the **HH** package in R. We also have a dashboard in Tableau.

Keywords: diverging stacked bar charts, graphics, population pyramids, Likert scales, **HH** package in R, psychometrics, **lattice** package in R, **vcd** package in R.

1. Ordered categorical scales, including rating scales

An ordered categorical scale is an ordered list of mutually exclusive terms. Ordered categorical scales are used, for example, in questionnaires where each respondent is asked to choose one of the terms as a response to each of a series of questions. The usual summary is a table that shows the number of respondents who chose each term for each question.

A rating scale is a form of psychometric scale commonly used in questionnaires. The most familiar rating scale is the Likert scale (Likert 1932), which consists of a discrete number of choices per question among the sequence: "Strongly Disagree", "Disagree", "No Opinion", "Agree", "Strongly Agree". Likert-type scales may use other sequences of bipolar adjectives: "Not Important" to "Very Important"; "Evil" to "Good". These scales sometimes have an odd number of levels, permitting a neutral choice. Sometimes they have an even number of levels, forcing the respondent to make a directional choice. Some ordered categorical scales

are uni-directional – age ranges or population quantiles, for example – for which negative and neutral interpretations are not meaningful.

For concreteness we present in Section 2 a dataset from a survey for which a natural display is a coordinated set of diverging stacked bar charts. We introduce the dataset by showing and discussing a multiple-panel plot of the entire dataset. Then we move to the construction of individual panels, and finally to the details of the coordinated presentation of the set of panels.

Our primary software tool is the `likert` function in the **HH** package (Heiberger 2014) in R (R Core Team 2013). The graphical presentation of Likert scale data in R can be used from the command line or from the **Rcmdr** menu (Fox 2005) using the **Rcmdr** plugin **RcmdrPlugin.HH** (Heiberger 2013). The **HH** package was originally designed as computing support for the book (Heiberger and Holland 2004). The code for all examples in this paper is available in R in three demonstration files included with the **HH** package. See `demo("likert-paper")` for the `formula` method into the underlying `lattice::barchart` plotting technology (Sarkar 2013, 2008). See `demo("likert-paper-noFormula")` for the same set of examples using the matrix and list of matrices methods. See `demo("likertMosaic-paper")` for the same set of examples using an alternate set of functions built on the `vcd::mosaic` as the underlying plotting technology (Meyer, Zeileis, and Hornik 2006, 2013; Zeileis, Meyer, and Hornik 2007).

We have also made available a workbook (Robbins 2013) in Tableau (Tableau Software 2011).

We follow this with illustrations of other types of data analysis situations, for ordered scales that are not rating scales, where diverging stacked bar charts are helpful.

Other graphical display techniques have been used for Likert-type datasets. We illustrate several and discuss why we believe the diverging stacked bar chart is better for this type of data.

The remainder of the paper is organised as follows: Section 2 describes the dataset we will use in many of our plots and shows and discusses a multiple-panel graph of this dataset. Section 3 discusses and illustrates the construction of a single-panel graph in R. Section 4 details the graphic design and programming issues that arise when designing Likert-type plots. Section 5 applies the general programming issues to the specifics of programming in R. Section 6 gives a collection of examples of multiple-panel plots for many different types of ordered categorical scales. Section 7 gives examples with other types of categorical scales. Section 8 shows several types of plots that we think do not work well when used for ordered categorical scales. Section 9 discusses the construction of diverging stacked barcharts based on the `mosaic` function in the **vcd** package in R. Section 10 illustrates the construction of diverging stacked barcharts in the software environment of Tableau. Section 11 provides our conclusions.

2. Display of professional challenges dataset

Our primary data example is from an *Amstat News* article (Luo and Keyes 2005) reporting on survey responses to a question on job satisfaction. A total of 565 respondents replied to the survey. Each person answered one of five levels of agreement or disagreement with the question "Is your job professionally challenging?" The respondents were partitioned into non-overlapping subsets by several different criteria. For each of the criteria, the original authors were interested in comparing the percent agreement by that criterion's groups.

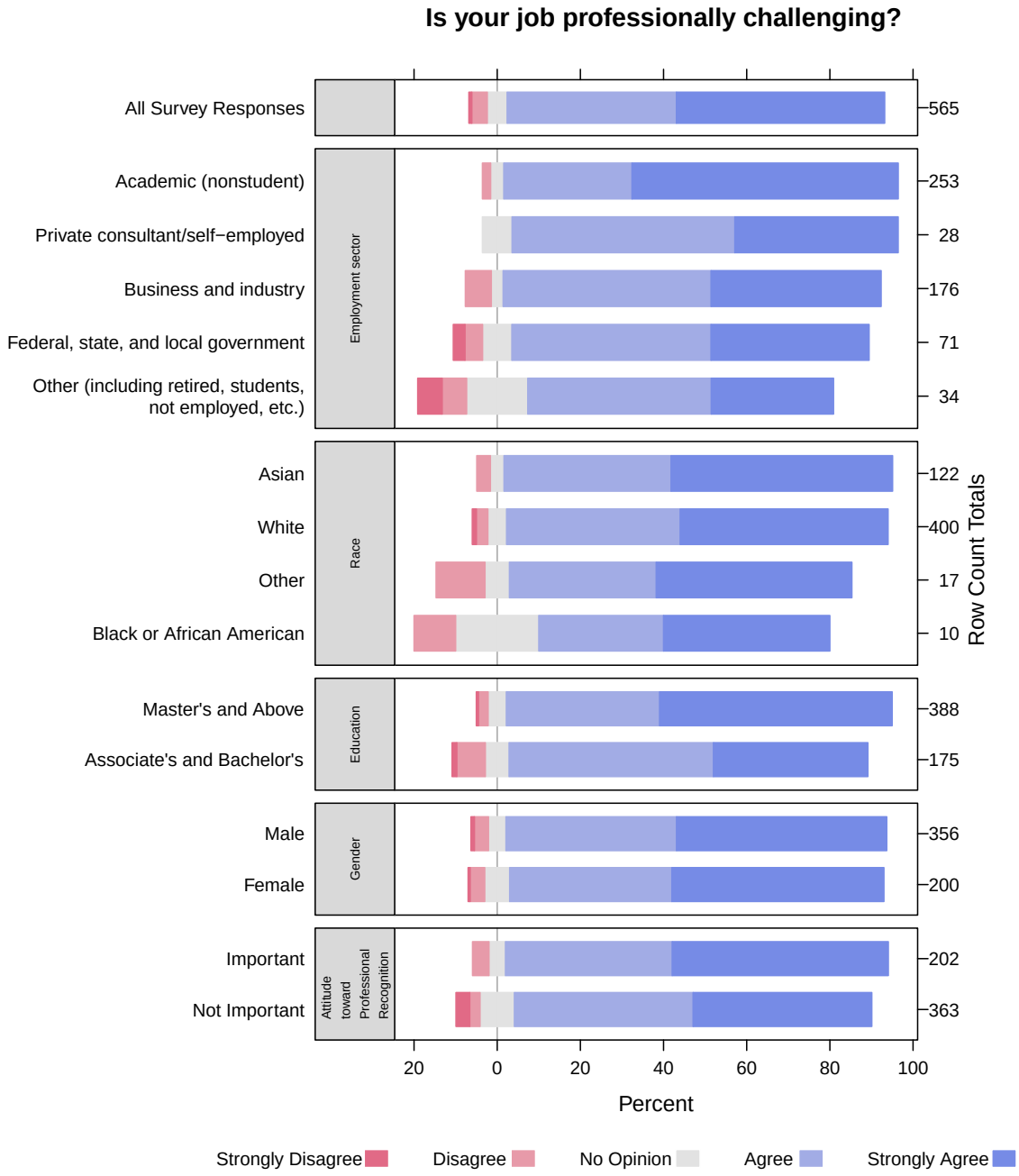

Figure 1: Survey responses to a question on job satisfaction (Luo and Keyes 2005). A total of 565 respondents replied to the survey. Each person answered one of five levels of agreement or disagreement with the question "Is your job professionally challenging?" Each panel of the plot shows a breakdown of the respondents into categories defined by the criterion listed in its left strip label.

In Figure 1, we show the complete results of the survey as a coordinated set of diverging stacked bar charts. In this section we concentrate on the appearance of the plot for its function of representing the meaning of the dataset. In Section 4, we will look at the compu-

tational details for construction of the plot. The calling sequence with the R function `likert` used to create the plot in this figure, along with the plots for all other figures, is shown in `demo("likert-paper")`.

There are six panels in the plot. The top panel shows "All Survey Responses". The remaining panels show different partitions of the 565 respondents. In the second panel from the top, for example, the criterion name "Employment sector" is in the left strip label. The respondents self-identify to one of the five employment groups named in the left tick labels. The number of people in each group is indicated as the right-tick label. Each stacked bar is 100% wide. Each is partitioned by the percent of that employment group who have selected the agreement level indicated in the legend below the body of the plot. The legend is ordered by the values of the labels. Darker colors indicate stronger agreement. Gray indicates the neutral position, in this example, "No Opinion". The bar for the neutral position is split, half to the left side of the vertical zero reference line and half to the right side. The reference line is placed behind the bars to prevent it from artificially splitting the neutral bar into two pieces. The default color palette has red on the left for disagreement and blue on the right for agreement. See Section 3 for a discussion of color palettes.

The intent of this plot is to compare percents within subgroups of the survey population; consequently we made all bars have equal vertical thickness. The panel heights are proportional to the number of bars in the panel. The x-axis labels are displayed with positive numbers on both sides. The bars within each panel have been sorted by the percent agreeing (totaled over all levels of agreement). We usually prefer horizontal bars, as shown here, because the group labels and the names of the groups are easier to read when they are displayed horizontally on the y-axis.

3. Single-panel displays

In this section, we look at the data for just the employment panel of the full plot in Figure 2. Table 1 shows the respondents divided into five employment categories and the counts for each agreement level within each employment category. In Section 4 we discuss the tasks involved in constructing a single-panel display. In Section 6, we discuss the combination of the individual panels into a multiple-panel plot.

	Strongly Disagree	Disagree	No Opinion	Agree	Strongly Agree
Employment Sector					
Academic (nonstudent)	0	5	8	78	162
Business and industry	0	11	5	88	72
Federal, state, an local government	2	3	5	34	27
Private consultant/self-employed	0	0	2	15	11
Other (including retired, students, not employed, etc.)	2	2	5	15	10

Table 1: The respondents have been divided into five employment categories. The rows (employment categories) are displayed in the original order: alphabetical plus other. Columns are displayed sequentially, with disagreement to the left and agreement to the right.

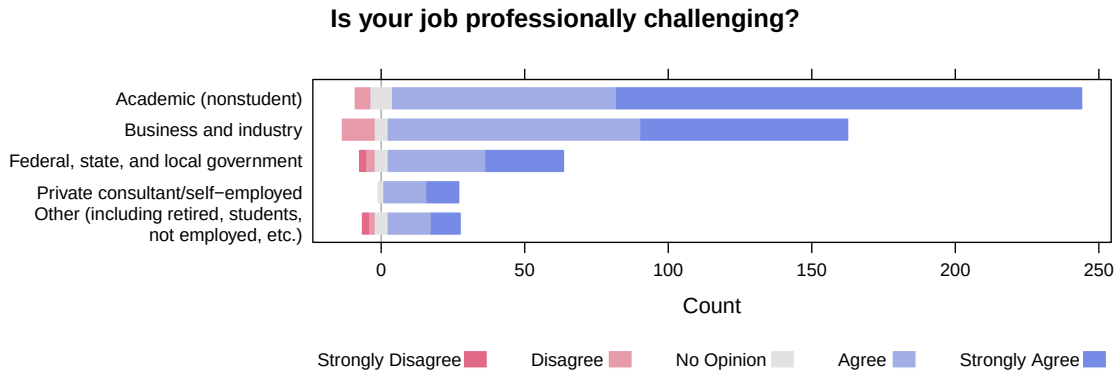

Figure 2: This plot is a direct translation of the numerical values in Table 1 to graphical form. Blue is agree, red is disagree, gray is no opinion. The strongest message in this presentation is that the sample has a very large percentage of academics.

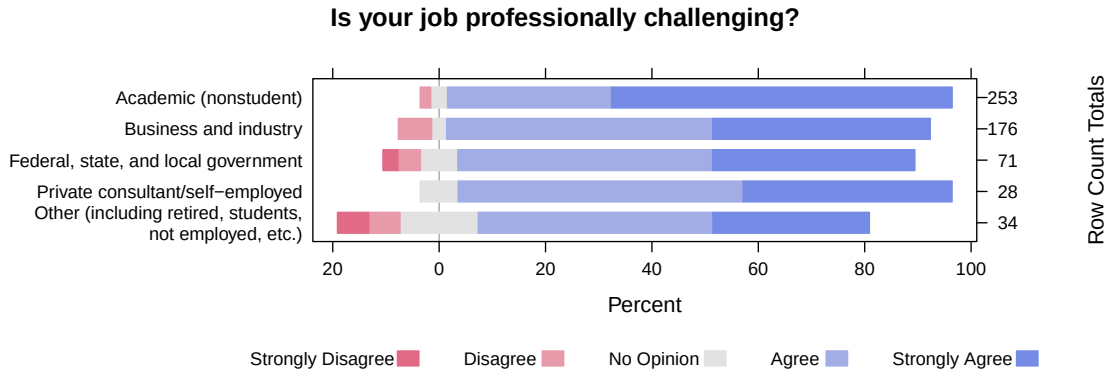

Figure 3: In this variant of the plot, we display the row percents. We do not want to lose information about the uneven selection of respondents from the employment sectors so we display the counts as the right axis labels. Now we see that "Academic (nonstudent)" stands out as the largest percentage of dark blue on the graph.

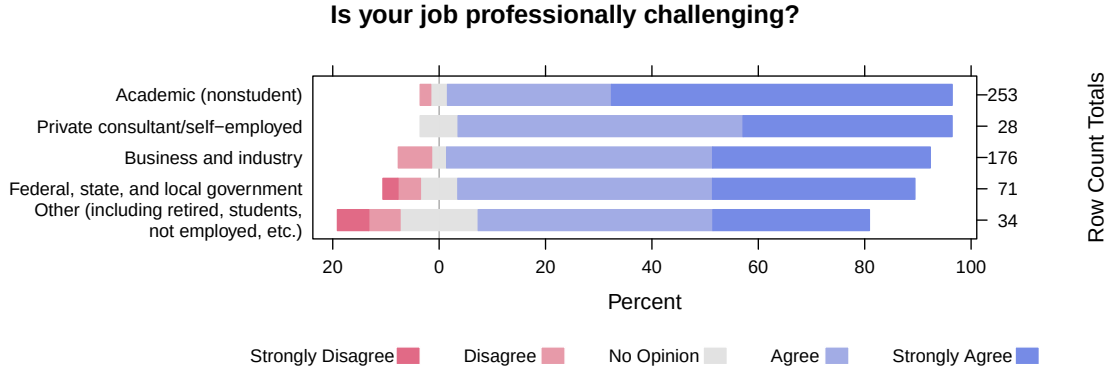

Figure 4: In our third presentation of the same table, we now sort the rows of the table by the total percent agree (dark blue + light blue + $\frac{1}{2}$ gray). Data-dependent ordering is usually more meaningful than alphabetical ordering for unordered categories.

Diverging stacked bar charts are easily constructed from Likert scale data. Each row of the table is mapped to a stacked bar in a bar chart. Usually the bars are horizontal. The counts (or percentages) of respondents on each row who agree with the statement are shown to the right of the zero line in one color; the counts (or percentages) who disagree are shown to the left in a different color. Agreement levels are coded from light (for closer to neutral) to dark (for more distant from neutral). The counts (or percentages) for respondents who neither agree nor disagree are split down the middle and are shown in a neutral color. The neutral category is omitted when the scale has an even number of choices. Our default color palette is the (red–blue) palette constructed by the `diverge_hcl` function in the **colorspace** package (Ihaka, Murrell, Hornik, Fisher, and Zeileis 2013; Zeileis, Hornik, and Murrell 2009) in R. The colors in the diverging palettes have equal intensity for equal distances from the center. The base colors red and blue have been chosen to avoid ambiguity for those with the most prevalent forms of color vision deficiencies.

It is difficult to compare lengths without a common baseline; see pages 54–57 of Robbins (2005, reissued 2013) and the reference therein to Cleveland and McGill (1984). We are primarily interested in the total count (or percent) to the right or left of the zero line; the breakdown into strongly or not is of lesser interest so that the primary comparisons do have a common baseline of zero.

Figure 2 shows a direct translation of the counts in Table 1 to a plot. The strongest message in this presentation is that the sample has a very large percentage of academics. It is harder to compare relative proportions in the employment categories because the total counts in each row are quite disparate.

Figure 3 displays the percents within each row. Now it is easy to see that a large majority of the people in all employment categories have a positive answer to the survey question. We do not want to lose the disparity in row totals, so we use the row count totals as the right-axis tick labels.

For plots such as Figure 3 with a single panel, and also for multiple-panel plots where the rows are distinct in each panel, we can still do better. Figure 4 shows the same scaling as Figure 3, but this time the row order is data-dependent. Rows are now ordered by the total percent of positive responses. This allows the reader to recognize groupings among rows (in this example, groupings of employment categories) that show similar responses. Figure 4 improves on our previous figures in Robbins (2011b) and Robbins and Heiberger (2011) because it adds the row count totals as right-axis labels.

4. Design of diverging stacked bar charts

There are many tasks involved in the construction of Figure 1. We name and discuss the tasks here. Our recommended default orientation is to display the bars horizontally (see the first item in this list for why). Our discussion here uses both terms “rows” and “bars” to refer to the individual questions of a questionnaire. Similarly we use the terms “columns” and “levels of agreement” to refer to the set of possible responses to each question. We discuss programming these tasks in later sections of this paper. Section 5 discusses our programming of these tasks for `barchart` in R. Section 9 discusses our programming of these tasks for `mosaic` in R. Section 10 outlines our solution in Tableau.

Overall structure:

Orientation (horizontal or vertical bars): Usually the questions or group definitions will be several words long. Placing them horizontally along the y-axis makes them easy to read. The count or percent scale on the x-axis usually contains numbers with few characters and will therefore also be legible.

Choice of unit (usually percent or count): When all groups have the same count, for example when each row is a different question asked of the same group of people, the plot will be similar for either response. When the groups are defined by partitioning a population, the plots will differ depending on the response value, and it may be helpful to display the number of people on each row along with the percent distribution in the graph.

Display counts: When groups have different counts and percents are plotted, it is often helpful to have the counts displayed. We normally do so on the right tick labels.

Main title: Choose wisely. Focus on the message you want the reader to take away from the graph.

Subtitle: We have used subtitles for citation information or to direct the reader to some aspect of the data interpretation. We think of subtitles as an abbreviated caption.

Legend (horizontal or vertical, sequence, reverse, title, location): We normally order the legend in the same direction as the stacked bars. We place horizontal legends below the graph and vertical legends to the right of the graph. The sequence of bars in the legend should be the same as the sequence in the plot.

Color palette: See Section 3 for a more complete discussion. We default to the default (red–blue) palette constructed by the `diverge_hcl` function in the **colorspace** package. The user of our software has complete control over the color selection.

Alignment position: Usually we place agreement on the right and disagreement on the left, with neutral split evenly between the two sides. We offer an option to align the bars for different groups on any or between any two agreement levels.

Tick labels: We use the left tick labels for the question, the right tick labels for the total row count (in percent plots), and the bottom tick labels for the count or percent scale.

Construct each panel:

Order the columns: There is a natural order for Likert-type scales.

Handle the "neutral" column: We default to splitting the neutral column, by placing half its count or percent on the agree side and the other half on the disagree side.

Order the rows: There are two natural orders. When plotting results of a questionnaire, the questions are usually numbered and it is appropriate to retain the order implied by those numbers. More generally, by default we retain the order of the input dataset. When plotting a grouping of respondents, we often find the graphs

are easiest to read when the groups are ordered by length of the "agree" bar. We therefore provide an option (`positive.order = TRUE` in R) to order the rows by total count or row percentage (consistent with the choice of units) of the "agree" bar. A third order, alphabetical order, is usually not optimal. The user can specify alphabetical order, or any other order, by ordering the rows of the dataset before calling the `likert` function.

Bar heights (thickness): By default we use constant thickness over all bars. There are situations, particularly in single-panel displays, where the thickness might be data dependent. We use data-dependent thickness in the bars in Figures 12 and 13. In that way both percent and count are represented graphically.

Borders on color bars: We made the borders the same color as the body of the bars. The imperative is that it is the only way we found to prevent the split neutral response from appearing visually as two separate half-width bars. More generally, we found the redundancy of the borders provided interference with the message.

Join the panels together:

Panel layout: Multiple-panel displays can be arranged as rows of panels when there are subsets of questions that need to be identified (as in Figure 1), columns of panels (as in Figure 8), or both – rows for multiple subsets of questions and columns for either multiple sets of respondents or for multiple presentations of the same data (as in Figure 6 with both counts and percents).

Panel heights: When the groupings do not have the same number of levels, for example the criteria partitions in Figure 1, the panels need to have heights proportional to the number of levels, and the individual bars were set to have a constant height in all panels.

Panel widths: Columns of panels might not have equal width, for example in Figure 6 we show both counts and percents for the professional challenges example. We use most of the area of the graph for the percents, and a much narrower area for the counts.

Orientation (horizontal or vertical panels): When the groupings are different, as in Figure 1, vertical placement of panels works best. In this case the x-axis is common across all panels and can be shared. When the groupings are the same, as in Figures 8 and 9, horizontal panels work well, because the y-labels can be shared.

Panel labels (number of rows in label, rotation): Vertical arrangement of panels suggests left strips for the criteria labels. In Figure 1, we found 90° rotation of long labels split into several lines, and with a slightly smaller font, worked well. In Figure 8, horizontal labels in a top strip worked well. In the population pyramid in Figure 10, where the two panels are thought of as the left and right sides of a single panel, we find that a pair of top axis labels works well.

Animation: The five panels in Figure 9 could be displayed in a slide show as a sequence of single-panel slides and we could watch the population shift. For that type of animation, it would probably be better to have a full set of years rather than the ten-year snapshots we used here.

Labeling conventions: Population pyramids such as Figure 10 are common in sociology. They have the convention of a single centered y-axis.

5. Programming the diverging stacked bar chart

We have programmed our design for the diverging stacked bar charts as a plotting method in the **HH** package in R and as a worksheet in Tableau. We actually have two distinct implementations in R, `likert` based on the **lattice** `barchart` function and `likertMosaic` based on the **vcd** `mosaic` function. Both can produce all graphs displayed in this paper. Both depend on the `as.likert` function, to be described in Section 5.3, to map the data from the display convention of a table of numbers to the conventions required by the `barchart` or `mosaic` functions.

At this writing we recommend the `likert` implementation for most purposes. The `likert` function inherits from **lattice** both row and column labeling of panels for conditioned formulas, and complete axis annotation. Most of the discussion in this paper, and the rest of the discussion in this section, will be on the `likert` function.

The `likertMosaic` function, based on the `mosaic` function, has limited labeling of panels and axes due to limitations in the underlying `strucplot` framework in the **vcd** package. We describe the `mosaic` implementation in Section 9.

We describe the Tableau implementation in Section 10.

5.1. `likert` based on `barchart`

The primary function `likert`, an alias for both `likertplot` and `plot.likert` (the naming convention is discussed in Section 5.6), is a generic function into a set of S3 methods which depend on the structure of the first argument. We recommend using the `formula` method, with a model `formula` as the first argument and with a data.frame as the second argument. The `formula` method does some mild rearrangement of the data based on the specifications of the formula and then forwards the call to the `barchart` function in the **lattice** package. Multiple conditioning factors may be specified in the formula. The formula specification (with optional labeling arguments not shown) for Figure 1 is

```
R> library("HH")
R> data("ProfChal", package = "HH")
R> likert(Question ~ . | Subtable,  data = ProfChal,
+    as.percent = TRUE, positive.order = TRUE,
+    scales = list(y = list(relation = "free")), layout = c(1, 6))
```

The `.` in the formula is expanded into the symbolic sum of all numeric variables in the data.frame, in this example to

```
R> likert(Question ~ "Strongly Disagree" + Disagree + "No Opinion" + Agree +
+    "Strongly Agree" | Subtable, data = ProfChal, as.percent = TRUE,
+    positive.order = TRUE, scales = list(y = list(relation = "free")),
+    layout = c(1, 6))
```

The levels of the condition factors define and name the panels. All our examples can be constructed directly from a data.frame with the `formula` method. Internally, the `.`, or equivalently the symbolic sum, is converted to a new factor and used as the `groups` argument to `barchart`. `barchart` uses the levels of the `groups` factor to construct the stacks and to assign the colors.

We also have S3 methods for matrix, data.frame, table, array, and list input. The code for these methods constructs each panel separately and then uses functions from the **latticeExtra** package (Sarkar and Andrews 2013) to combine them into a single '`trellis`' object. To use the list specification for Figure 1, we must first convert the data.frame into a list of matrices, one per subtable.

```
R> tmp <- data.matrix(ProfChal[, 1:5])
R> rownames(tmp) <- ProfChal$Question
R> ProfChal.list <- split.data.frame(tmp, ProfChal$Subtable)
```

We then operate directly on the list.

```
R> likert(ProfChal.list, as.percent = TRUE, positive.order = TRUE)
```

The list method for `likert` sends the matrix in each list item to the matrix method for `likert`, which calls `barchart` for the actual plotting, and stores the resulting '`trellis`' object in a list of '`trellis`' objects. The individual '`trellis`' objects are combined with `c.trellis` into a single multiple-panel '`trellis`' object. The names of the list items in the argument become the names of the panels of the '`trellis`' object. The number of rows in each list item becomes the argument to the `resizePanels` function.

The levels of the condition factors for the data.frame method, or equivalently, the items in the list for the list method, define the panels in a multiple-panel display. All panels in a multiple-panel display must have the same columns. They do not need to have the same rows; the `ProfChal` example has different rows (sets of questions) in each panel. They may have the same rows, as in the `SFF8121` example in Figure 8.

Each panel of the graph is based on a matrix or table of rows (usually groupings of respondents for a single question, or separate questions by the same set of respondents) by columns (levels of agreement). Each panel is a diverging stacked bar chart constructed by using the `barchart` function in the **lattice** package.

The **HH** package also provides `plot.likert` S3 methods for more complex data structures. We can plot a two-way structure stored as a matrix, as the numerical columns in a data.frame, as a table, as a `ftable`, and as a `structable`. We can plot a k-dimensional array or table interpreted as a set of two-way tables. We can plot a list of two-way structures with the same number of columns but different numbers of rows in each, for an example see Figure 1. We can take multiple single-panel charts and combine them into a single multiple-panel plot with coordinated axes using functions in the **latticeExtra** package.

By default, the rows within each table in an array are plotted in their original order to simplify comparison between the table and the plot. For consistency, we also default to the original row order with other data structures. We frequently want to order the rows by the counts (or percentages) who agree. This makes most sense for single-panel plots or for multiple-panel plots with different rows labels in each panel (again see Figure 1).

It is possible to produce a Likert plot with a list of objects with different numbers of columns, although not with the list method of `plot.likert`. These must be done manually by using the `ResizeEtc` function (based on `resizePanels` and other functions in the **latticeExtra** package) on each of the individual Likert plots. They cannot be constructed automatically because the legend is based on the number of columns in the last item in the list and will have the wrong number of values for some of the panels. Since it is not systematic, it requires user intervention. We illustrate the sequence in Section 6.1.

5.2. Use of S3 methods in R implementation

The methods for `likert` are built on S3 methods which depend on the structure of the argument.

With a `formula` object as the first argument and with a data.frame as the second argument, the data is restructured to a data.frame with all numerical values in one column and with the names of the numerical columns as levels of a newly constructed grouping factor. Conditioning factors, if any, define the rows and columns of the '`trellis`' structure. Within each panel, the rows of the original data.frame become the rows of the display. The constructed data.frame is sent to `barchart` for plotting.

For other classes of objects as the first argument, the call is dispatched to the appropriate method of `likert`. For matrix and data.frame objects as arguments, the call to `likert` is dispatched to the default method – a call to the **lattice** `barchart` function. The resulting '`trellis`' object is usually displayed on screen or printed. Table and vector arguments are converted to matrices. Lists of matrices with a common set of columns are maintained as lists. k-dimensional arrays are converted to $(k-2)$-dimensional lists of consistently structured (common set of rows and columns) matrices. Each element of a list of matrices is graphed with the **lattice** `barchart` function, and the resulting '`trellis`' objects are stored in a list with the same structure. The list is processed into a single complex '`trellis`' object with our `ResizeEtc` function.

5.3. `as.likert` function

We are using the `barchart` in the **lattice** package as our workhorse function in R. Mapping the conventions of Likert-style tabular data to the conventions of `barchart` in order to produce diverging stacked bar charts is the hardest and most critical part of the programming.

The **lattice** `barchart` function is designed to draw bars either positive or negative from zero. Therefore we need to restructure the tables of counts from the original matrix of positive counts to a form that will allow `barchart` to draw the graphs we need. We need to make the counts for the "Disagree" categories negative, and split the "No Opinion" category into two groups, each half as wide as the original, one positive and one negative.

The convention used by `barchart` for the stacking order of negative values is different than the order used in tables of Likert-type data. Therefore we need to change the order. The mechanics of the changes needed are encoded in the `as.likert` function (discussed below) called internal to the default `plot.likert` method. The original and revised orders are illustrated here:

Original order:

```
R> tmp <- ProfChal[2:6, 1:5]
```

```
R> rownames(tmp) <- substring(ProfChal$Question[2:6], 1, 5)
R> tmp
```

```
      Strongly Disagree Disagree No Opinion Agree Strongly Agree
Acade                 0        5          8    78            162
Busin                 0       11          5    88             72
Feder                 2        3          5    34             27
Priva                 0        0          2    15             11
Other                 2        2          5    15             10
```

Revised order: The "No Opinion" column is split into two. Columns on the "Disagree" side are given negative values. The columns are resequenced to put the column containing "Disagree" values prior to the column containing "Strongly Disagree". The `"positive.order"` attribute stores the sort order that will be used for the rows if the user specifies `positive.order = TRUE`.

```
R> as.likert(tmp)
```

```
      No Opinion Disagree Strongly Disagree No Opinion Agree Strongly Agree
Acade       -4.0       -5                 0        4.0    78            162
Busin       -2.5      -11                 0        2.5    88             72
Feder       -2.5       -3                -2        2.5    34             27
Priva       -1.0        0                 0        1.0    15             11
Other       -2.5       -2                -2        2.5    15             10
attr(,"color.seq")
[1] 3 2 1 3 4 5
attr(,"positive.order")
[1] 4 5 3 2 1
attr(,"nlevels")
[1] 5
attr(,"original.levels")
[1] "Strongly Disagree" "Disagree"          "No Opinion"
[4] "Agree"             "Strongly Agree"
attr(,"xlimEqualLeftRight")
[1] FALSE
attr(,"xTickLabelsPositive")
[1] TRUE
attr(,"class")
[1] "likert" "matrix"
```

The function `HH:::as.likert.simplified.odd` illustrated here is the central section of the method `HH:::as.likert.matrix`. This simplified function takes a matrix argument with the same number of positive and negative levels and with a neutral level (thus an odd number of levels).

```
R> as.likert.simplified.odd <- function(x, nc = ncol(x),                ##  1
+    colorset = (1:nc) - (nc + 1)/2) {                                  ##  2
```

```
+    ind.neg <- rev((1:nc)[colorset < 0])                          ##  3
+    ind.pos <- (1:nc)[colorset > 0]                               ##  4
+    ind.zero <- (1:nc)[colorset == 0]                             ##  5
+    x <- cbind(-x[, ind.zero, drop = FALSE]/2,                    ##  6
+      -x[, ind.neg, drop = FALSE],                                ##  7
+       x[, ind.zero, drop = FALSE]/2,                             ##  8
+       x[, ind.pos, drop = FALSE])                                ##  9
+    attr(x, "color.seq") <- c(ind.zero, ind.neg, ind.pos)         ## 10
+    pos.columns <- seq(to = ncol(x),                              ## 11
+      length = length(c(ind.zero, ind.pos)))                      ## 12
+    attr(x, "positive.order") <- order(apply(x[, pos.columns,     ## 13
+      drop = FALSE], 1, sum))                                     ## 14
+    x                                                             ## 15
+ }                                                                ## 16
R> HH:::as.likert.simplified.odd(tmp)
```

The variable `colorset` in line 2 holds the conversion of the (positive) column numbers [1, 2, 3, 4, 5] to ordered numbers [−2, −1, 0, 1, 2] representing the ordered column labels ("Strongly Disagree", "Disagree", "No Opinion", "Agree", "Strongly Agree"). The variables in lines 3–5 are indices into the column numbers that will be used to reorder the columns to the sequence used by the `barchart` function. Lines 6–9 do the actual reordering. Lines 6 and 8 split the neutral column's counts into two columns, one on the positive side and one on the negative side. The `"color.seq"` attribute in line 10 orders the colors to the new column sequence. The `"positive.order"` attribute holds the row sequence that will order the rows by the total length of the `"Agree"` bar (the sum of the "Strongly Agree", "Agree", and half of the "No Opinion" values).

The complete `as.likert` generic function (the function that was simplified to the above example) produces an object with additional attributes. The `"nlevels"` is the number of original (before splitting the "No Opinion" group) levels of the response variable. The `"original.levels"` are the labels of the original levels of the response variable. The attribute `"xlimEqualLeftRight"` records the input argument choice to enforce symmetric positive and negative x limits. The attribute `"xTickLabelsPositive"` records the input argument choice to display counts or percents on the "Disagree" side of the plot with positive numbers. The `"class"` is the result of prepending the value '`likert`' to the class of the input data object.

5.4. Specification of methods

The calling sequence we recommend for specifying the diverging stacked bar charts is the `formula` method

```
likert(Question ~ . | Subtable, data = dataframe)
```

where `.` is interpreted as the symbolic sum of all numeric variable names. We also can use the generic function directly on an ordinary matrix (or data.frame, table, array, or list) of counts

```
likert(regularobject)
```

In order to use this calling sequence with S3 methods, we defined `likert` to be an alias for `plot.likert`, thus

```
plot.likert(regularobject)
```

is equivalent, and we then let S3 handle everything.

The function `plot.likert` serves a dual role. It is both a method for `plot` and the generic starting point for the `plot.likert.xxxx` methods.

With S3 dispatch technology for the `plot.likert` generic, the S3 dispatcher looks at the `class` of the argument and then dispatches the appropriate `plot.likert.xxxx` method as discussed in Section 5.2. The name `plot.likert` implies, correctly, that this is the `plot` method for '`likert`' objects.

There is a '`likert`' class of objects. '`likert`' objects are constructed with the `as.likert` function illustrated in Section 5.3. The '`likert`' object is difficult to read. Not until we get to the workhorse functions, either `plot.likert.default` or `plot.likert.formula`, do we actually need the `as.likert` construct or the '`likert`'-class objects.

Because `plot.likert` is a method for `plot`, the calling sequence using the '`likert`' class of objects

```
likertobject <- as.likert(regularobject)
plot(likertobject)
```

also works. We recommend the simple

```
likert(regularobject)
```

5.5. Variable-width bars

In `barchart` the `box.width` argument applies to what are effectively the columns in the original Likert-style data. Thus all `"Agrees"` will have one width and all `"Disagrees"` will have a different width. There is no control of the width of an entire stacked bar. To display what we think of as a single-panel Likert graph whose bars have variable width, see for example Figure 12, we must program it as a multiple-panel display. We use `resizePanels` to control the heights of the panels, and we draw the panel borders in `"transparent"` color. Extending this to a multiple-panel graph constructed with a conditioning factor is possible.

5.6. Naming of the function

The function name `likert` is an alias for both `likertplot` and `plot.likert`. We chose this naming for two distinct reasons; each alias represents one of those reasons.

We recommend the formula method as the primary way to specify the diverging stacked barchart. The notation for the formula method is an exact parallel to the similar notations for other **lattice** functions:

```
R> tmp <- data.frame(y = factor(LETTERS[1:5]), a = 1:5, b = 6:10, c = 11:15)
R> tmp.matrix <- data.matrix(tmp[, 2:4])
```

```
R> row.names(tmp.matrix) <- tmp[, 1]
R> likert(y ~ a + b + c, data = tmp)
R> likertplot(y ~ a + b + c, data = tmp)
R> xyplot(y ~ a + b + c, data = tmp)
R> bwplot(y ~ a + b + c, data = tmp)
R> dotplot(y ~ a + b + c, data = tmp)
R> stripplot(y ~ a + b + c, data = tmp)
R> plot.likert(y ~ a + b + c, data = tmp)
```

We think the name `likertplot` is too long for comfortable typing, and see no ambiguity in using the shorter alias `likert`. The `plot.likert` name, discussed in the next paragraph, also works with the formula method.

We initially designed the `likert` function as a matrix method and extended it to array, list, data.frame, and table methods. In this setting we thought of the result of the `as.likert` function as an object of class '`likert`'. We therefore wrote the `plot.likert` plotting method for '`likert`' objects. The '`likert`' objects are almost unreadable by people (see the display in Section 5.3) and using them in code leads to ugly statements like

```
R> plot(as.likert(tmp.matrix))
```

It is much easier for people when we compact the nested function calls to the simpler, and sufficient, `likert(tmp.matrix)`. The next three statements are equivalent. The fourth is not, because it dispatches it to the matrix method of `plot`.

```
R> plot.likert(tmp.matrix)
R> likert(tmp.matrix)
R> likertplot(tmp.matrix)
R> plot(tmp.matrix)
```

The function `plot.likert` is both a method for `plot` for '`likert`' objects and a generic function with methods for formula, data.frame, list, table, and array. The `likertplot` name from the previous paragraph also works here. Please note that the really simple statement `plot(tmp.matrix)` cannot be used to dispatch a Likert plot. The matrix method of `plot` would be dispatched instead.

We constructed the `likertMosaic` as a generic function with methods to work exactly the same as the `likert` generic function.

6. Multiple-panel displays

Our illustration in Figures 2–4 is a single panel. It displays information on a single question for a partition of the respondents into several groups based on employment. Figure 1 is a multiple-panel display containing Figure 4 combined with other partitions of the same set of respondents.

Section 6.1 discusses the construction of multiple-panel plots. The remaining subsections in Section 6 give examples of the use of multiple-panel plots.

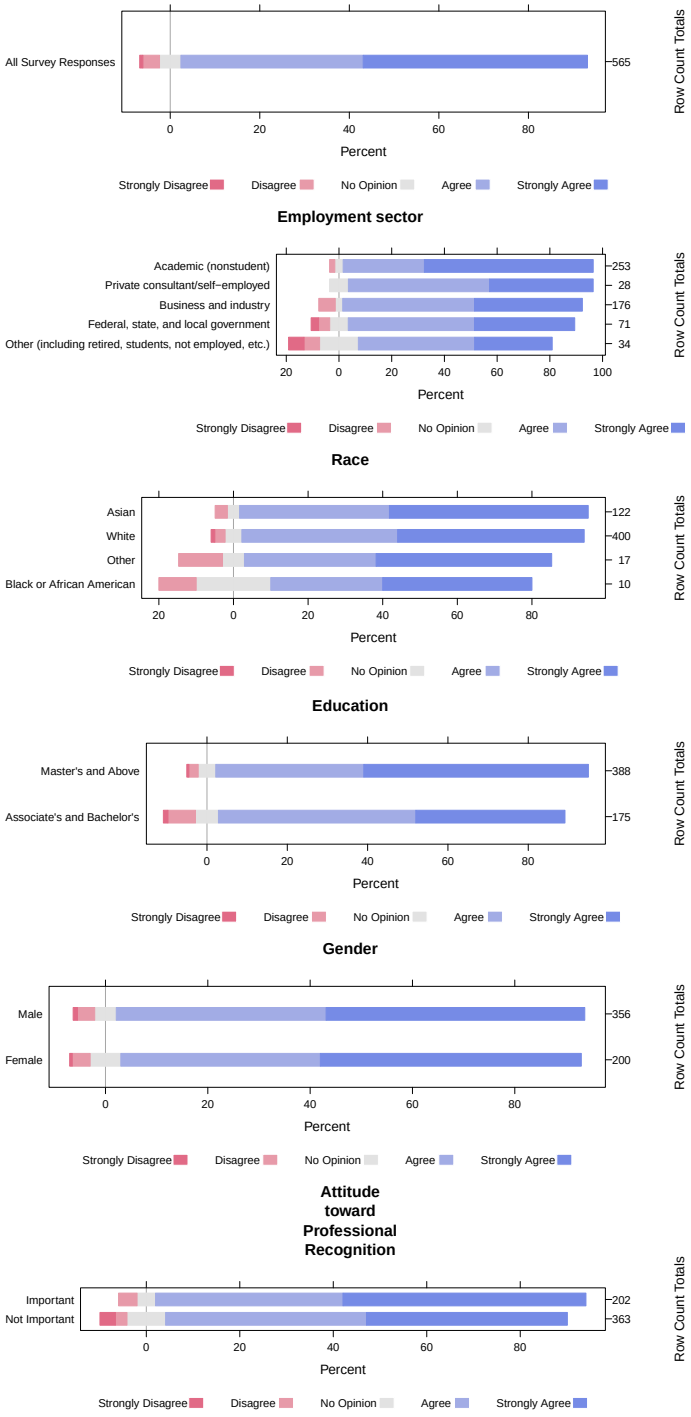

Figure 5: Six independent plots, one for each characteristic used to subset the respondents. Because each characteristic has a different set of rownames, and each rowname has a different number of characters, the panels do not have the same width. Since the bars in each panel have different ranges, the axes are not identical. There is no alignment across panels. Figure 1 shows the result of using the `resizePanels` function to combine these panels into a single multiple-panel plot with coordinated axes.

6.1. Constructing multiple-panel graphs in lattice in the R language

Multiple-panel diverging stacked bar charts are produced directly by specifications using the `formula` method when there is a conditioning factor in the formula.

They can be constructed manually from separate plots with the `resizePanels` function. Figure 5 shows independent plots (`AA` ... `FF`) of the six panels of Figure 1. Each of the panels has a different range on the x-scale (the percent scale). Each of the panels has a different number of rows and different labels for the rows. In order to combine the panels into a single coordinated plot, we must force a common x-range on all panels, and arrange for the heights of each panel to be proportional to the number of rows in that panel. We need to use the `resizePanels` function in the **latticeExtra** package for automatic coordination of the axes and spacing of the panels. The principal function call is

```
resizePanels(h = c(1, 5, 4, 2, 2, 2),
  c(AA, BB, CC, DD, EE, FF), layout = c(1, 6), x.same = TRUE))
```

This call, together with the additional calls executed by `demo("likert-paper")` and `demo("likert-paper-noFormula")`, will reproduce Figure 1 with fully coordinated axes and equal row spacing.

The `resizePanels` call is incorporated into the method for lists of tables with equal numbers of columns. The technique can be used manually for lists with unequal numbers of columns. We cannot do the unequal-columns case automatically because there is no simple way to determine the right matching of the legend to the column colors.

6.2. One question with multiple subsets of the sample

Figure 1 shows responses to the same question for the same population of respondents partitioned into several series of groups based on additional characteristics.

The partitions have different numbers of groups. In order to retain the same vertical spacing between parallel bars, the vertical space allocated for the panels must differ. In R we constructed a function that uses the capabilities of the **latticeExtra** package, primarily the `resizePanels` function, to control spacing of the panels. The different panels have been labeled by the name of the partitioning characteristic. In this example the panels are identified by a left strip label. Within each panel the bars have been sorted by the total percent of positive responses.

6.3. Multiple subsets of questions with multiple responses

Figure 6 shows both percents and counts for the same set of responses to the same set of question.

6.4. One or more subpopulations with multiple questions

Figure 7 is differently structured. The data are the responses to a survey sponsored by the New Zealand Ministry of Research Science and Technology (New Zealand Ministry of Research Science and Technology 2006). Here we have two different sets of questions that have been asked of the same set of respondents. Sometimes the respondents can be subdivided. If we

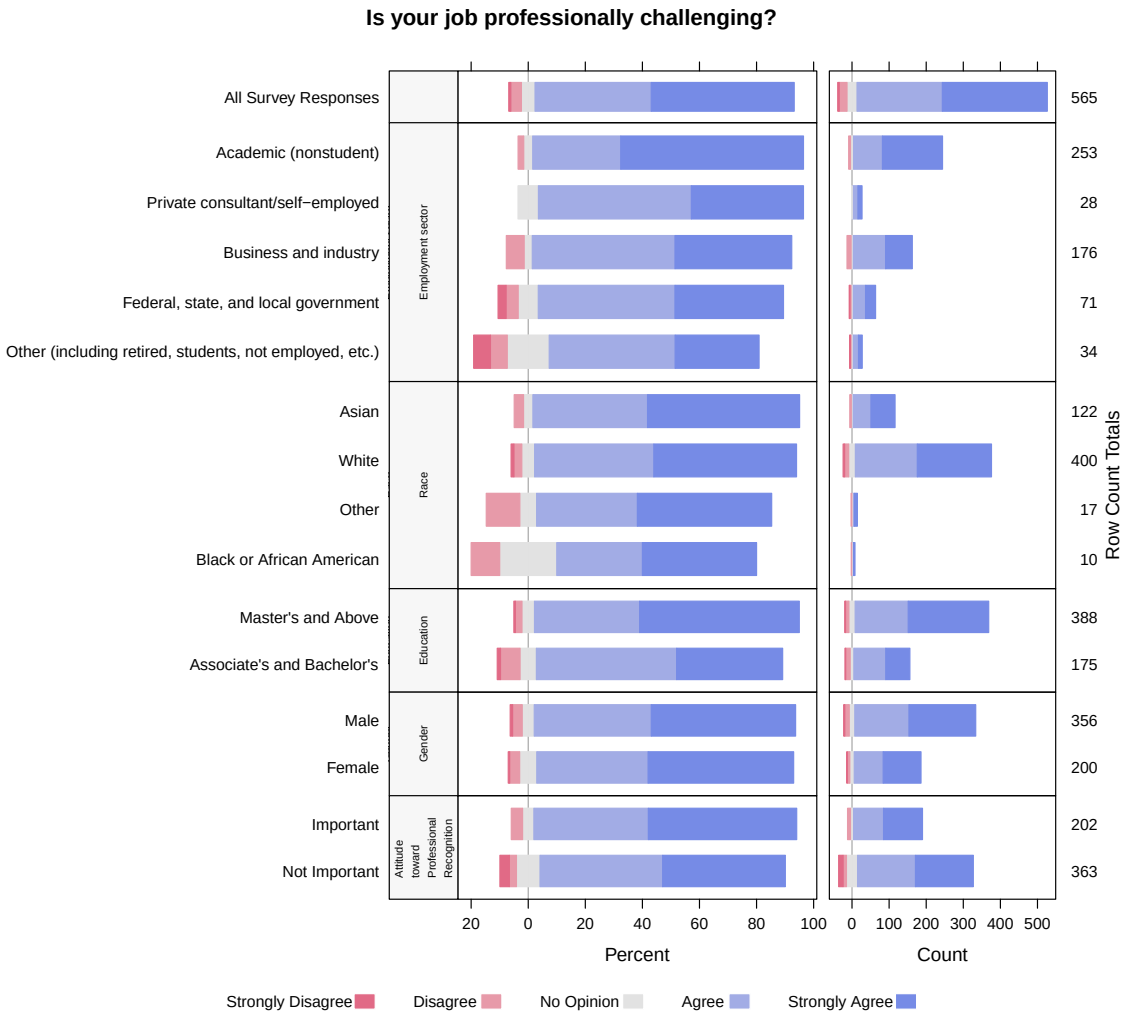

Figure 6: Both percents and counts for the professional challenges data. The left set of panels is identical to Figure 1. The right set is the same data as counts.

had the data separately for the male and female subsets, we could have a plot similar to Figure 7 but with two columns of panels, left and right, one for each subset.

6.5. Multiple sets of respondents: Student feedback forms

Temple University requires student evaluations of courses at the end of each semester. Figure 8 shows the results of the Student Feedback Form for Heiberger's Statistical Computing course, and compares it to the average of all graduate courses in the university for that semester.

In this example we chose to keep the original order of the questions because the intent is to use this as both a self-evaluation tool and as a management tool. In practice, there will be thousands of these graphs, one for each section of each course every semester. Currently the results are released by the Provost's office as tables to individual faculty members, to department chairs, and to Deans. They are much easier to read as graphs.

Individual faculty members could look at all their courses per semester, or over a period

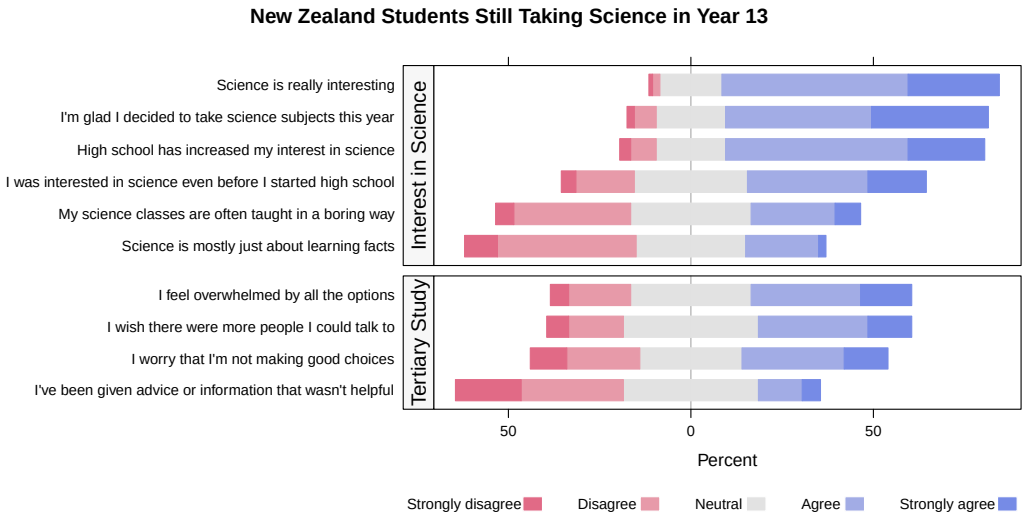

Figure 7: Two sets of questions have been asked of all respondents. Each set is presented in its own panel.

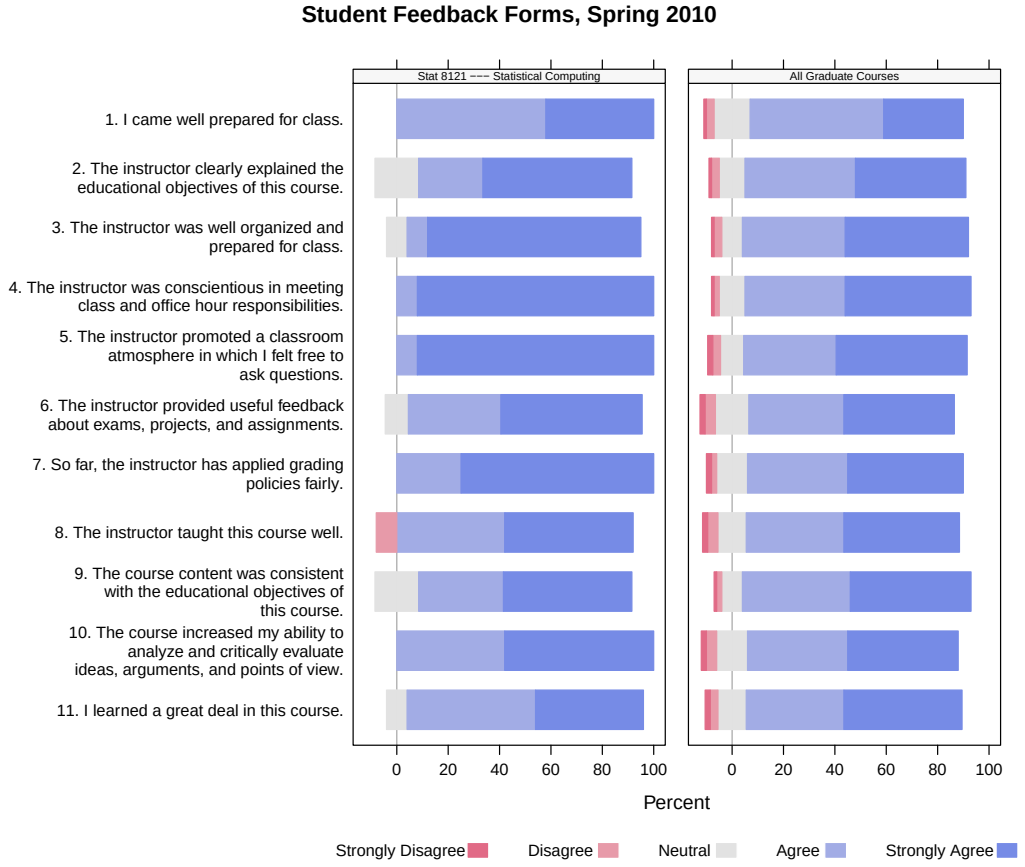

Figure 8: Student Feedback Forms for Stat 8121 Statistical Computing compared to the combined results from all graduate courses in the entire university.

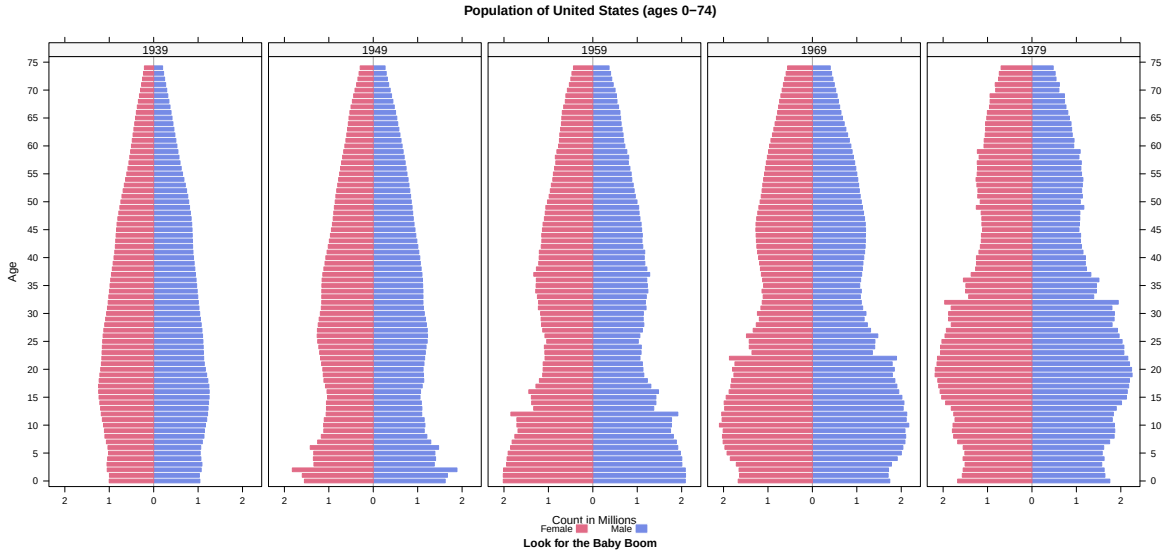

Figure 9: Five population pyramids at ten-year intervals years 1939–1979. We can see the baby boom start at the bottom of the population graph for 1949 and work its way up over time. We have placed the age tick labels on both the left and right axes of the set of panels.

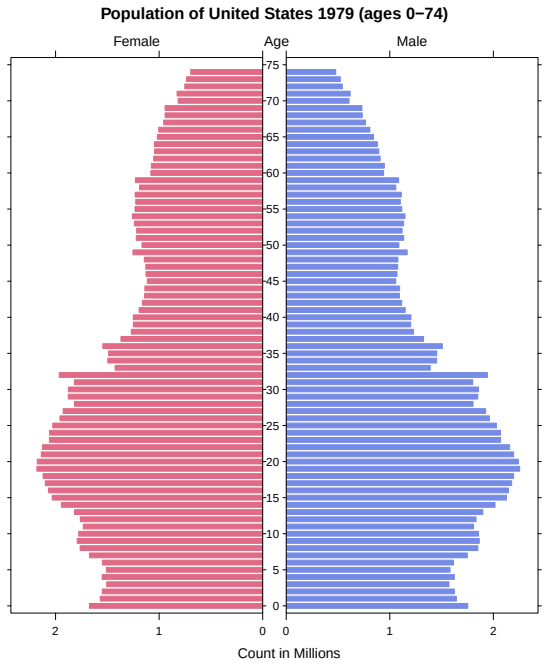

Figure 10: USA population in 1979. We have placed the age tick labels in the center, the position traditionally used for population pyramids.

of several semesters. Chairs would look at all sections of each course in their department. The university provides reference information by course, by department, by school, for the whole university, and by level (graduate, undergraduate first two years, undergraduate last two years).

6.6. Common structure at multiple times: Population pyramids

Population pyramids are used in demographic studies and in epidemiological studies. The pyramid is a pair of back-to-back bar charts, one for males and one for females. We display the population pyramid as a Likert-type scale with two levels, male and female, for each age range. We show two figures here. Figure 9 shows five pyramids at ten-year intervals years 1939–1979, with the y-labels on both the left and right axes. Figure 10 shows the USA population in 1979 in the standard presentation with the y-labels in the center. The data is from the `USAage` dataset in the **latticeExtra** package.

7. Examples: Other scales

7.1. Financial data over time

Diverging stacked bar charts have been used in many types of literature for a long time.

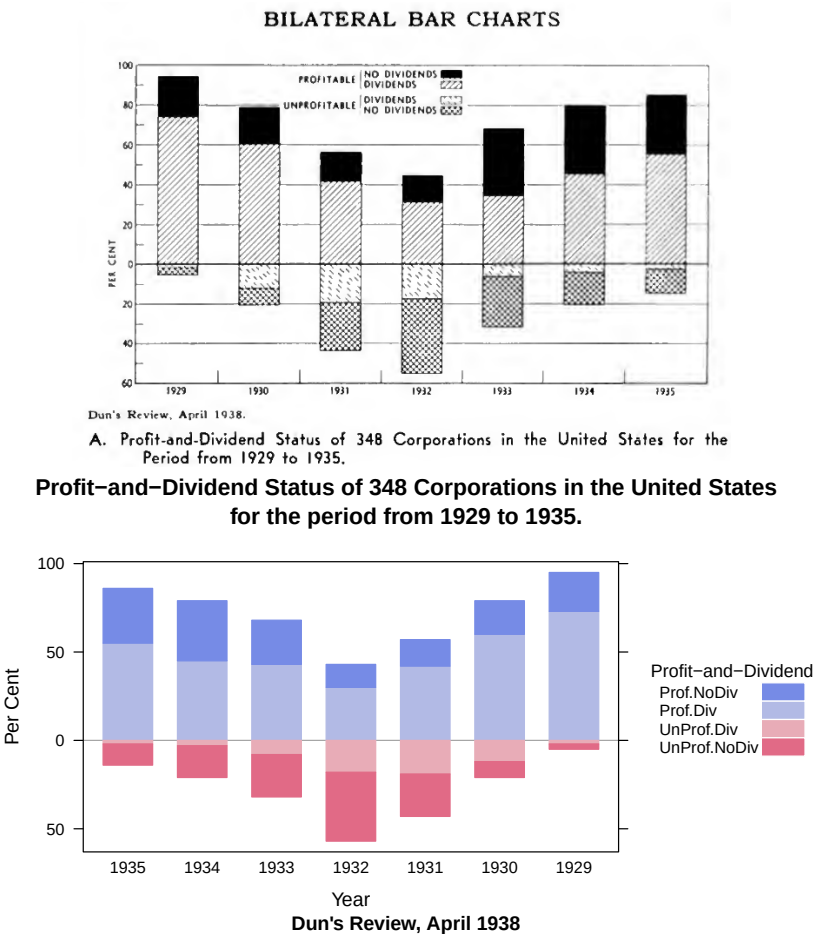

Figure 11: The original on the top is from Brinton (1939). Our version was drawn in 2011. We believe it would make more sense to interchange the two "Profit" bars and then put the dark blue color on the "Prof–Div" status. We chose to maintain the order used in the original.

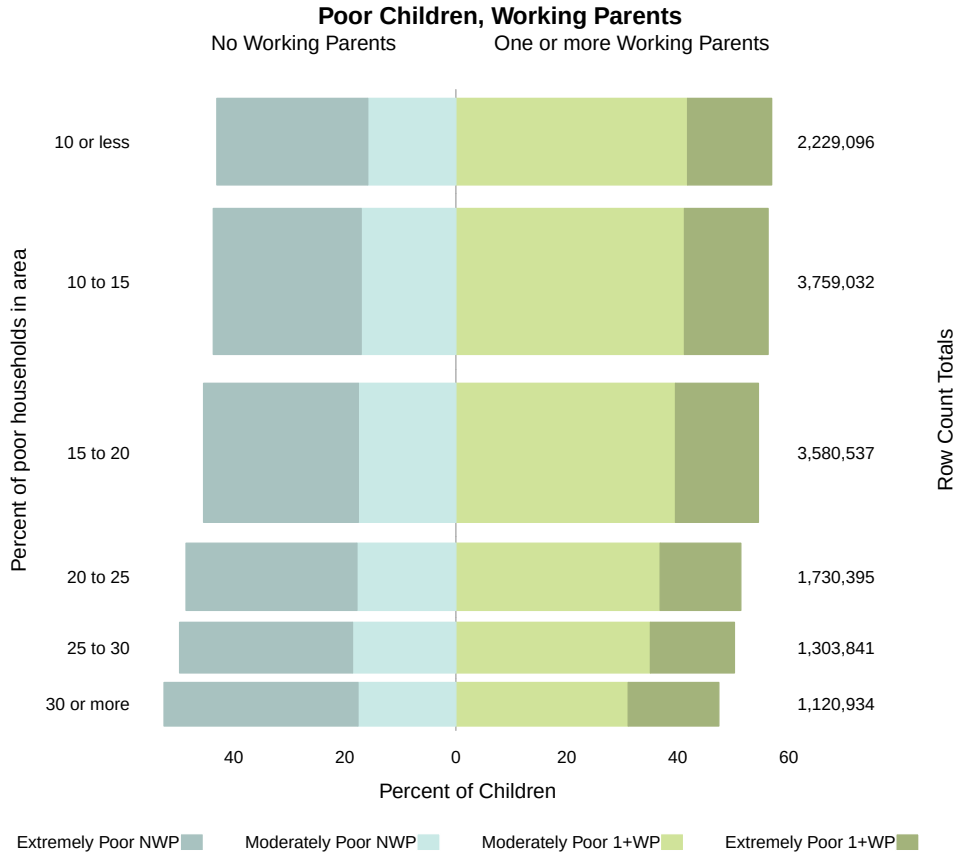

Figure 12: Bars have variable heights, proportional to the number of children in the areas they represent. Each bar shows by its color split the proportions of extremely and moderately poor children by the number of working parents.

Figure 11 shows a plot of Profit-and-Dividend status. The original is from Brinton (1939). Our version was drawn in 2011. The scale is really a 2×2 crossing of two factors even though it is displayed as a single 4-level factor.

7.2. Data informing economic policy

Presidential aspirant Newt Gingrich made the claim, "Really poor children in really poor neighborhoods have no habits of working and have nobody around them who works." This claim is easily refuted by looking at the data. Figure 12 comes from our revision (Heiberger and Robbins 2011) of a graph based on US Census data presented in a New York Times Op-Ed column by Charles Blow (Blow 2011).

Figure 12 uses the color scheme in Blow's original graph. The bars represent the poverty level of the areas (census areas of 100,000 people) in which the children live. The principal new graphic feature is that the heights of the bars, and therefore their areas, are proportional to the number of children they represent. The bars are all 100% wide. In this way, each bar simultaneously shows (a) the proportion of poor children by poverty status and working parent status, and (b) the number of children. It is clear that most poor children have at least

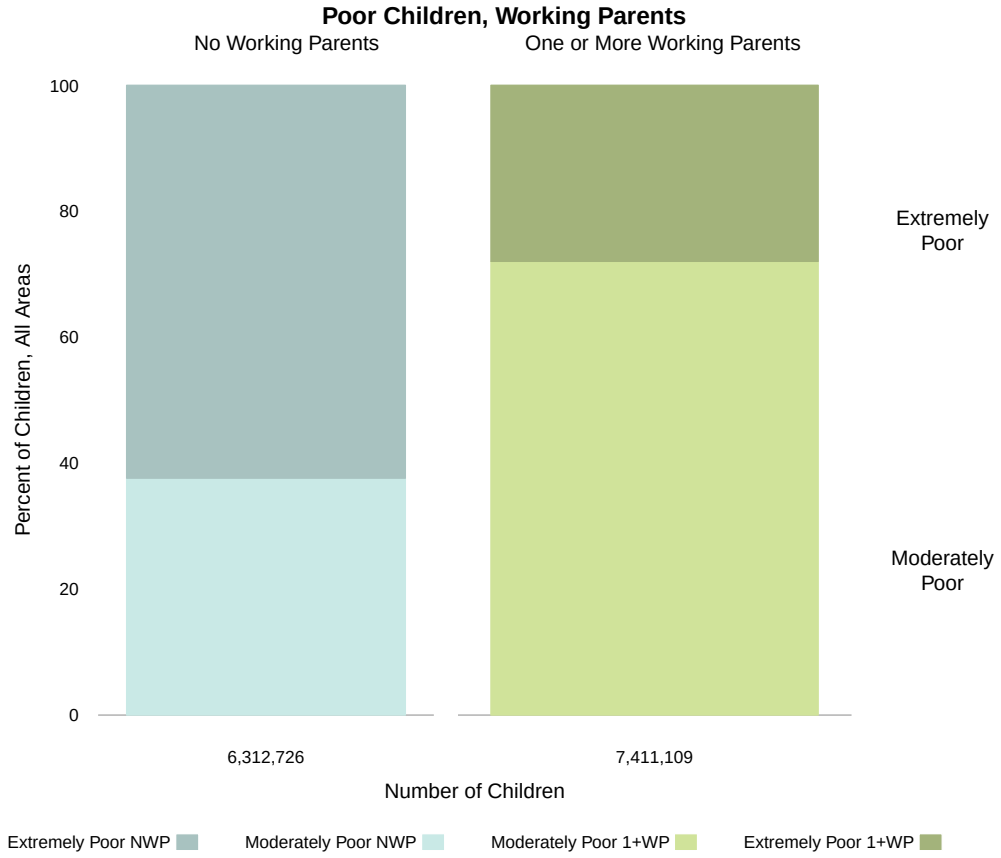

Figure 13: Summarized information from Figure 12. Bars have variable widths, proportional to the count of children they represent. Each bar shows by its color split the proportion of extremely poor children by the number of working parents.

one working parent. It is also clear that there is a shift to a higher percent of children with no working parents as the area poverty level goes up. In Figure 13 we aggregated all bars over the census areas and changed the orientation. Here the two principal messages are: (a) Most poor children have working parents (the green bar on the right is wider than the blue bar on the left), and (b) children with working parents have a smaller percent of extreme poverty than children without working parents (the darker green on the right is a smaller percent of the total green than the darker blue on the left of the total blue).

Our Figure 13 is consistent with a hypothesis that says creating jobs will decrease the number of households in the group 'households with no working parents' and increase the number in the group 'households with at least one working parent.'

7.3. One-directional scale: Age

Not all ordered scales are two-directional. Stacked bars can also be used to represent distributions along a one-dimensional ordered scale. Age, for example, is always positive and increases in a single direction. The example in Figure 14 (from Robbins 2011a) uses colors from a sequential palette for the age groups. We do not use a diverging palette that is appropriate

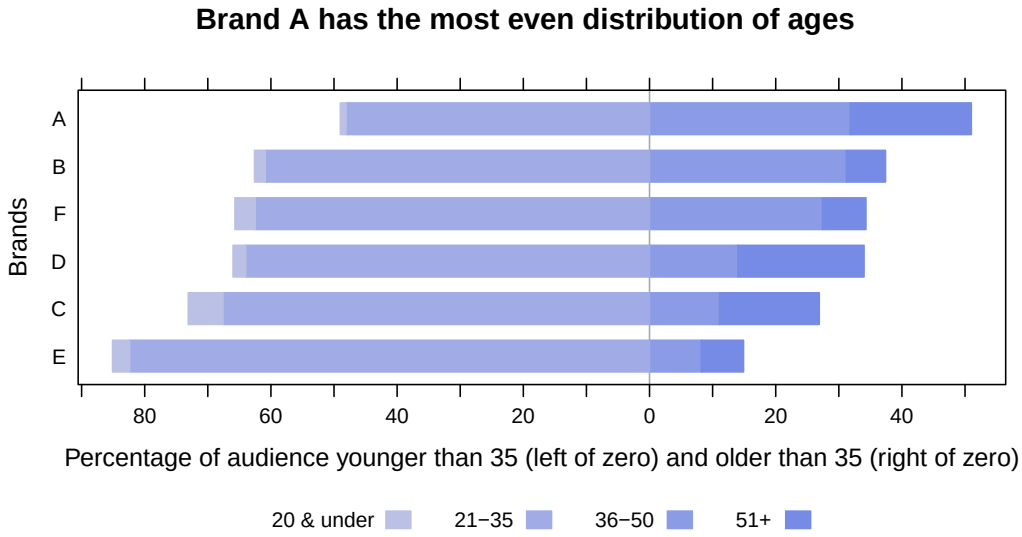

Figure 14: This data is from a current study (direct communication by author). The brands are ordered by percent positive. The study is intended to compare the percentage of the target audience younger than 35 to those older than 35. Therefore the common baseline for the bars is set to age 35.

when groups are defined by a characteristic, such as strength of agreement or disagreement, that can increase in two directions. For this example we chose to use the `col` argument directly and gave it an intermediate set of blue colors selected with the `sequential_hcl` function from the **colorspace** package. The intent of the study was to detect differences in brand preferences for audience members younger than 35 with those for audience members older than 35. Therefore we constructed the scale with the reference line for percents centered at age 35.

8. Alternate displays of Likert-like data

Figure 1 shows the Likert-scale results of a survey displayed using a coordinated set of diverging stacked bar charts. In Robbins and Heiberger (2011) we show a variety of other ways we have seen Likert scales presented, point out advantages of diverging stacked bar charts over other popular displays of Likert scales, and recommend diverging stacked bar charts. Here we show two other bar chart displays of Likert-scale data. We recommend neither. We use these displays to illustrate several of the design issues discussed in Section 4.

8.1. Grouped bar charts

A common display uses grouped bar charts. Figure 15 shows the data in Figure 2, the employment sector of Figure 1, plotted as a grouped bar chart.

Although the grouped bar chart is clear, accurate and easy to read, we think that it emphasizes the wrong comparisons. The grouped bar chart facilitates comparisons of the various levels of agreement within an employment sector. However, it is likely that the developers of the

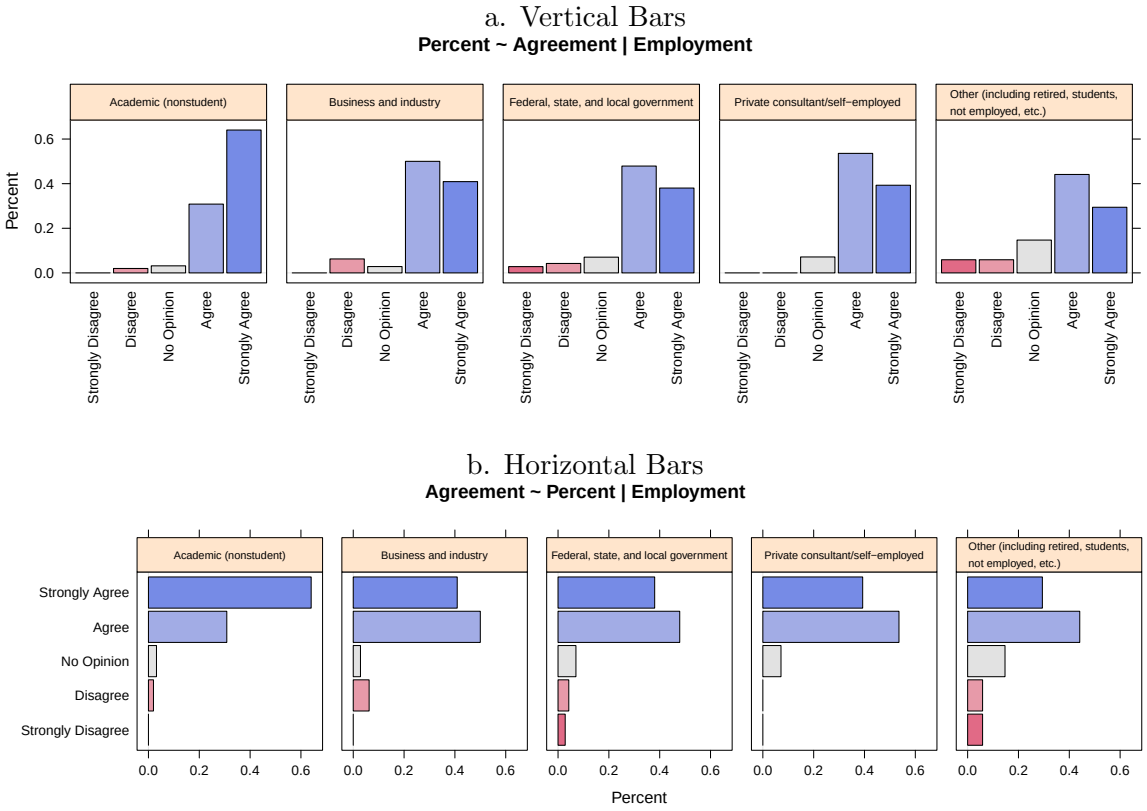

Figure 15: Not Recommended. Compare these plots with Figure 3. The employment sector of the professional challenges data is shown with a grouped bar chart, with groups defined by the types of employment. The top figure shows vertical bars; the bottom figure shows horizontal bars. This arrangement facilitates comparisons of the various levels of agreement within an employment sector. It does not facilitate comparing agreement levels across employment sectors. The vertical bars have multiple copies of the employment level names written vertically. Reading the names of the levels is difficult. The horizontal bars have the names written once on the left margin. The comparisons across employment levels within an agreement level are possible, though still not easy.

survey were more interested in knowing in which employment sectors the workers agreed that their work was professionally challenging and in which they disagreed. Figure 1 emphasizes this comparison of employment levels by placing the counts or percentages of agreement to the right of the origin and those of disagreement to the left.

8.2. Heatmaps

We show in Figure 16 a heatmap of the employment criterion of the professional challenges dataset. This heatmap can be drawn as if it were a stacked barchart with the reference line at the left edge. It does not work well for Likert-type data because aligning at the left edge hides the fundamental comparison of "Percent Agree" with "Percent Disagree". Aligning at the center of the neutral position as in Figure 4 emphasizes the fundamental comparison.

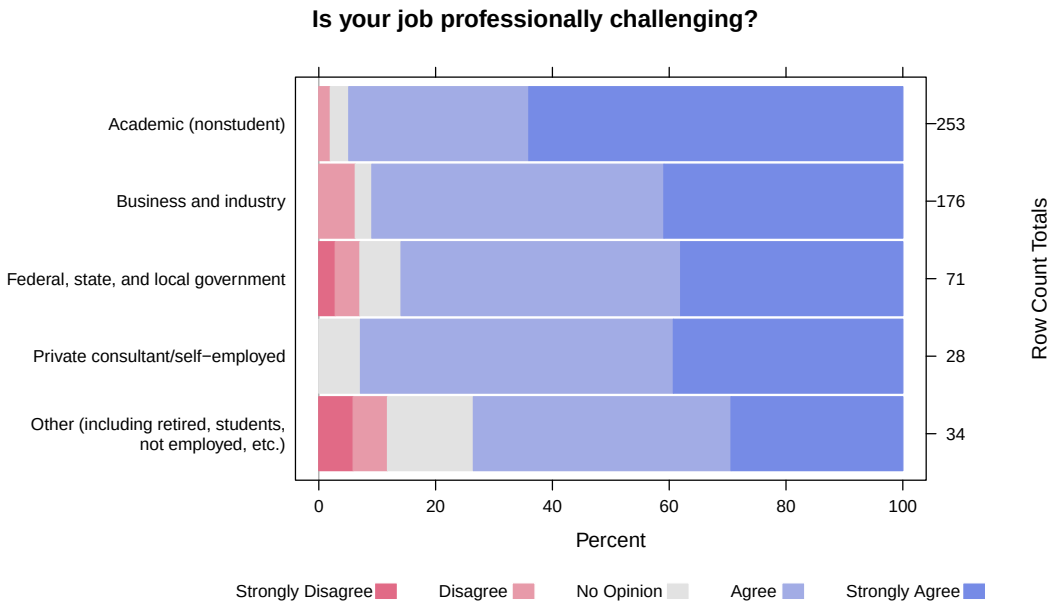

Figure 16: Not Recommended. Compare this plot with Figure 3. The employment sector of the professional challenges data is shown with a heat map, with groups defined by the types of employment.

9. `likertMosaic` based on `mosaic`

The primary function `likertMosaic` is a generic function into a set of S3 methods which depend on the structure of the first argument. We recommend using the `formula` method when it is applicable. All our examples work using at least one of the `likertMosaic` methods. Our current implementation of the `likertMosaic` function is limited to the annotation capabilities of the underlying `mosaic` function. Hence we do not have labeling of the numerical axes and we do not display the reference line at zero. On the positive side, it is much easier to specify variable-width bars (as in Figures 12 and 13) with the `likertMosaic` function than with the `likert` function. `likertMosaic` code for analogs of all figures in this paper is shown in `demo("likertMosaic-paper")`. Figure 17 shows the `likertMosaic` version of Figure 1.

9.1. `as.likert` function for `mosaic`

The Likert-style tabular data needs to be mapped to the `mosaic` conventions. The mapping is almost the same as the mapping to the `barchart` conventions. The critical component, splitting the neutral column, is identical. There are only two differences, and both are specified by changing the value of logical arguments to the `as.likert` function.

1. `mosaic` plots fill a rectangular area. Therefore `as.likert` constructs `"left padding"` and `"right padding"` data columns that bring the total count on each row to a constant. The two padding columns are assigned the `"transparent"` color for plotting. Their column names are not included in the legend.

2. `mosaic` plots the columns in the order of the data. Therefore, the reversal of the columns on the left hand side, as implemented in line 3 of the code on page 13, is not needed.

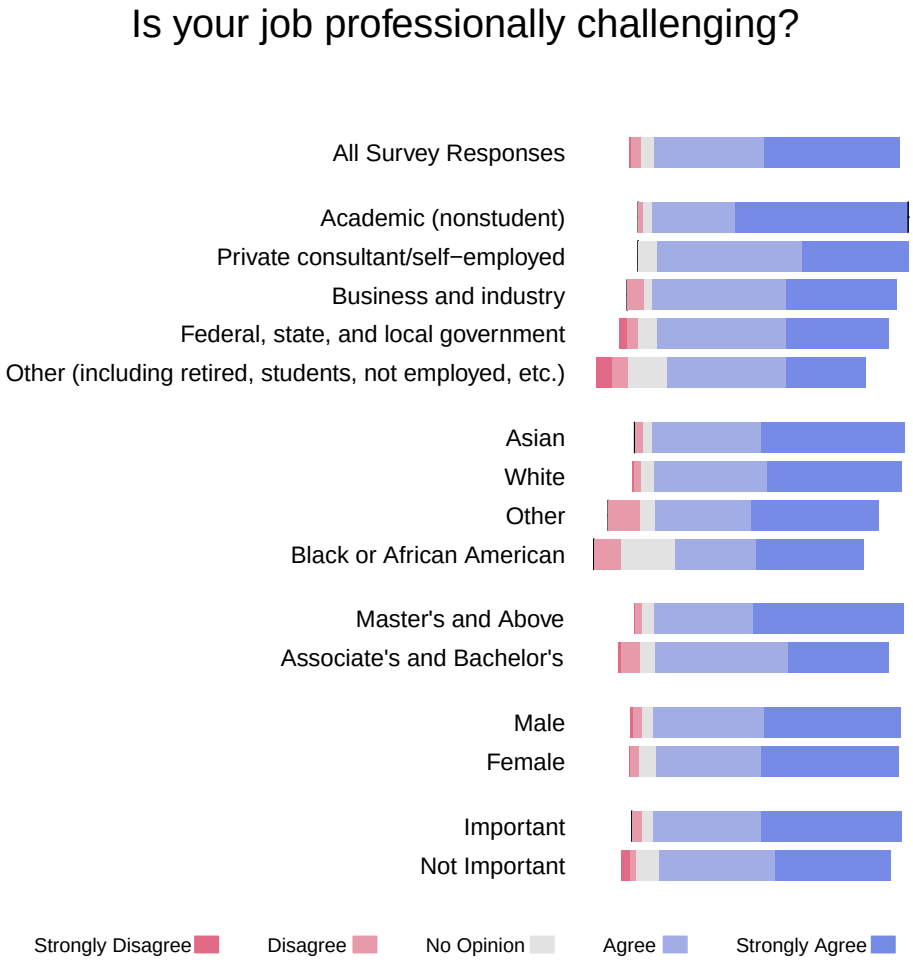

Figure 17: `likertMosaic` plot of survey responses to a question on job satisfaction (Luo and Keyes 2005). Compare this figure to Figure 1 drawn with the `likert` function. The `likertMosaic` function draws the tiles correctly, but it does not have some of the annotation – the group labeling, the numeric axis, and the reference line at zero – because it is restricted by the labeling capabilities of the underlying `strucplot` framework.

9.2. `likertMosaic` methods

`mosaic` is designed for plotting k-dimensional arrays. Therefore the array, matrix, list, and table methods essentially construct a new array with the extra columns on one dimension, and then forward it to `mosaic`. The `formula` method works a little harder. First, it needs to remove any columns from the input data.frame that are not included in the model formula. Second, if there are conditioning factors, it adjusts the vertical spacing to place additional space between sets of rows in different levels of the conditions. For the dimension representing the set of response levels, we specify that mosaic use zero spacing and no borders between tiles.

The formula specification for Figure 17, the analog of Figure 1, is

```
R> library("vcd")
R> likertMosaic(Question ~ . | Subtable, ProfChal, as.percent = TRUE,
+    positive.order = TRUE)
```

We have array, matrix and table methods for `likertMosaic`. The array method is the target method for which all other methods restructure their input data. We do not have a useful list method for `likertMosaic` because there is no way, analogous to the '`trellis`' object in **lattice**, to capture the graphic as an object. The list method applied to this example will draw six independently scaled graphs on six different pages, and there is no way to combine them into a single display.

9.3. Variable-width bars in `mosaic`

Variable-width bars are handled very differently in the `barchart` and `mosaic` implementations of the Likert plot.

In `mosaic` variable-width bars are natural. We construct them very simply: Rescale each row of the original data table to proportions by dividing by a scale factor, augment the rows with the left- and right-padding variables, then rescale again by multiplying by the same scale factor. The sums of the values in the adjusted rows, now including the padding, will be used by `mosaic` as the first conditioning factor and will have the intended width.

10. Diverging stacked bar graphs in Tableau

Diverging stacked bar charts can be created in many software environments. We illustrate in Figure 18 how to create them for the professional challenges data using the `Gantt Bar` chart in Tableau. The `Demographic` dimension refers to employment, race, etc.; `Group` refers

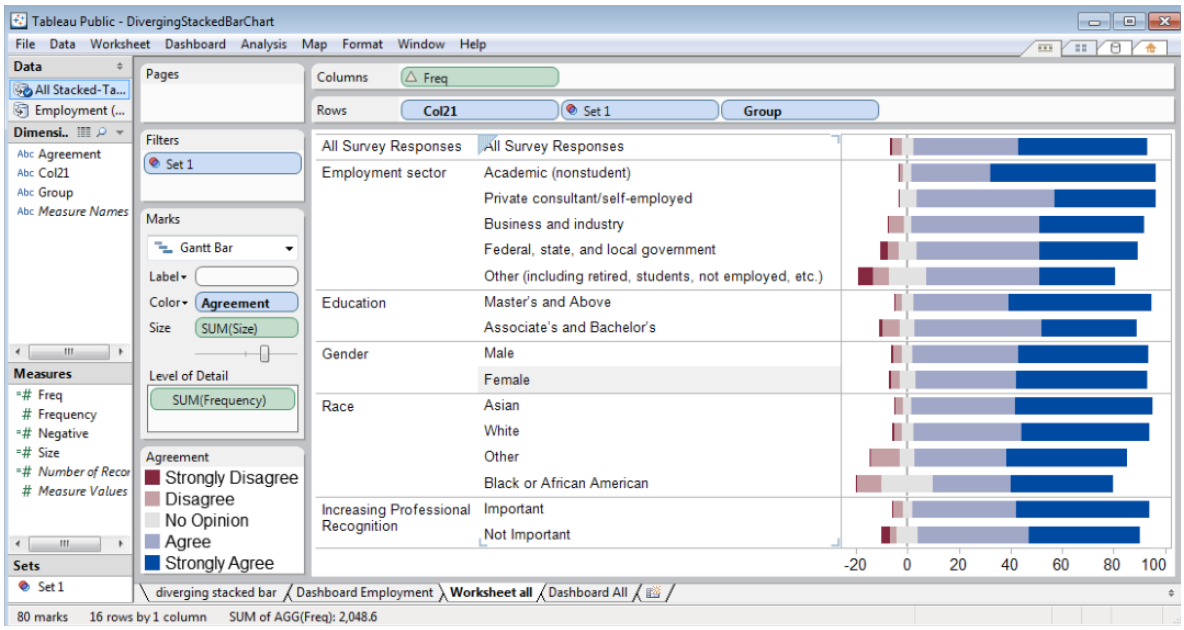

Figure 18: The Tableau worksheet that produced Figure 19.

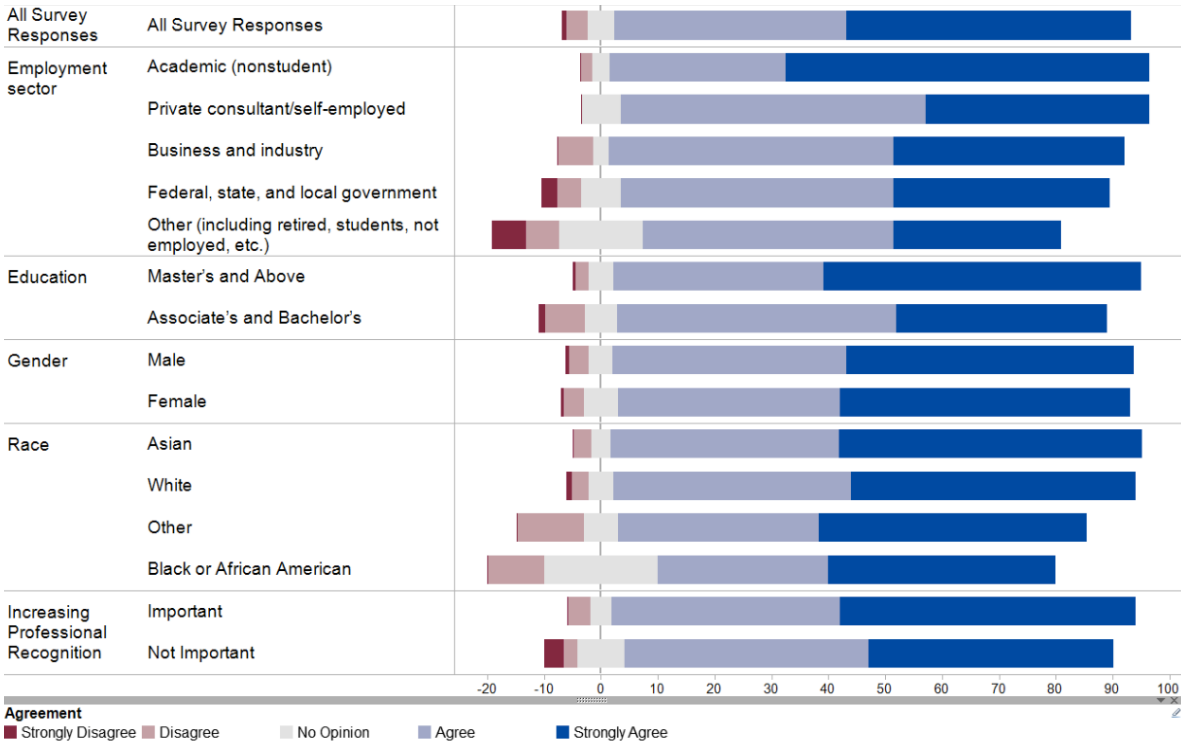

Figure 19: The professional challenges data in Figure 1 is redrawn in Tableau.

to the group within the demographic; and `Agreement` to the sequence "Strongly Disagree", "Disagree", "No Opinion", "Agree", "Strongly Agree". The measure `Frequency` gives the count or percent for the given `Demographic` and `Group`. `Frequency` needs to be on the `Level of Detail` shelf to include it in the tool tips. We need several calculated measures to make the same types of adjustments to the input table that we discussed and illustrated in Section 5.3 for the R calculations. The details are in the Tableau worksheet illustrated in Figure 18 and available for download at Robbins (2013). Figure 19 shows the final figure.

11. Conclusions

Diverging stacked bar charts provide an effective way to communicate summaries of data collected with Likert and other rating scales. We illustrate examples, using both single-panel and multiple-panel plots, for data from several fields. These charts are also useful for a number of applications in addition to rating scales, for example, age scales and population pyramids.

Diverging stacked bar charts are a useful addition to your graphical toolbox no matter what software you use.

The **HH** package for R includes a `likert` function to enable plotting of diverging stacked bar charts. It uses S3 methods to be responsive to the structure of the data. We recommend the formula method with the data stored in a data frame. We also provide methods for matrices (including tables and data frames), arrays, vectors, and lists of the above. The `likert` function is built on the `barchart` function in the **lattice** package. An intermediate function `as.likert` is used to rearrange the ordering of the categories from the conventions

of the Likert scale to the conventions of `barchart`. The **HH** package in R also includes a second implementation `likertMosaic` built on the `mosaic` function in **vcd**.

We also provide a dashboard (Robbins 2013) in Tableau using Gantt charts to illustrate how to create diverging stacked bar charts in Tableau.

The `likert` function in R was used for most of the figures in this paper.

Acknowledgments

Nick Cox alerted us to some of the early uses of diverging stacked bar charts. Section 10 benefited from discussions with Jock Mackinlay and Tim Lens. We had several e-mail discussions with Deepayan Sarkar on fine points of writing **lattice** panel functions. We have benefitted from comments and suggestions by the anonymous handling editor and referees. In particular, we wrote the formula method in response to their comments on the name of the function, and the `mosaic` implementation in response to their comments on variable-width bars. We have adopted many organizational rearrangements for the paper that they suggested.

References

Blow C (2011). "Newt's War on Poor Children." *New York Times*, p. A23. URL http://www.nytimes.com/2011/12/03/opinion/blow-newts-war-on-poor-children.html?_r=1.

Brinton WC (1939). *Graphic Presentation*. Brinton Associates. URL http://www.archive.org/details/graphicpresentat00brinrich.

Cleveland WS, McGill R (1984). "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods." *Journal of the American Statistical Association*, **79**, 531–554.

Fox J (2005). "The R Commander: A Basic Statistics Graphical User Interface to R." *Journal of Statistical Software*, **14**(9), 1–42. URL http://www.jstatsoft.org/v14/i09.

Heiberger R, Robbins N (2011). "Alternative to Charles Blow's Figure in "Newt's War on Poor Children"." *Forbes*. URL http://www.forbes.com/sites/naomirobbins/2011/12/20/alternative-to-charles-blows-figure-in-newts-war-on-poor-children-2/.

Heiberger RM (2013). ***RcmdrPlugin.HH**: **Rcmdr** Support for the **HH** Package*. R package version 1.1-40, URL http://CRAN.R-project.org/package=RcmdrPlugin.HH.

Heiberger RM (2014). ***HH**: Statistical Analysis and Data Display: Heiberger and Holland*. R package version 3.0-4, URL http://CRAN.R-project.org/package=HH.

Heiberger RM, Holland B (2004). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-PLUS, R, and SAS*. Springer-Verlag.

Ihaka R, Murrell P, Hornik K, Fisher JC, Zeileis A (2013). ***colorspace**: Color Space Manipulation*. R package version 1.2-4, URL http://CRAN.R-project.org/package=colorspace.

Likert R (1932). "A Technique for the Measurement of Attitudes." *Archives of Psychology*, **22**(140), 1–55.

Luo A, Keyes T (2005). "Second Set of Results in from the Career Track Member Survey." *Amstat News*, pp. 14–15.

Meyer D, Zeileis A, Hornik K (2006). "The Strucplot Framework: Visualizing Multi-Way Contingency Tables with **vcd**." *Journal of Statistical Software*, **17**(3), 1–48. URL http://www.jstatsoft.org/v17/i03/.

Meyer D, Zeileis A, Hornik K (2013). ***vcd**: Visualizing Categorical Data*. R package version 1.3-1, URL http://CRAN.R-project.org/package=vcd.

New Zealand Ministry of Research Science and Technology (2006). "Staying in Science." URL http://www.morst.govt.nz/Documents/publications/researchreports/Staying-in-Science-summary.pdf.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Robbins NB (2005, reissued 2013). *Creating More Effective Graphs*. Chart House, Ramsey, NJ. [Originally John Wiley & Sons].

Robbins NB (2011a). "Thinking Outside the Chart Menu." *Forbes*. URL http://www.forbes.com/sites/naomirobbins/2011/11/29/thinking-outside-the-chart-menu.

Robbins NB (2011b). "Visualizing Data: Challenges to Presentation of Quality Graphics—and Solutions." *Amstat News*, **411**, 28–30. URL http://magazine.amstat.org/blog/2011/09/01/visualizingdata.

Robbins NB (2013). "Dashboard for ProfChal Data." Tableau, URL http://public.tableausoftware.com/views/DivergingStackedBarChart/DashboardAll?:embed=y.

Robbins NB, Heiberger RM (2011). "Plotting Likert and Other Rating Scales." In *JSM Proceedings*, Section on Survey Research Methods, pp. 1058–1066. American Statistical Association, Alexandria, VA. URL https://www.amstat.org/membersonly/proceedings/2011/papers/300784_64164.pdf.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York. URL http://lmdvr.R-Forge.R-project.org.

Sarkar D (2013). ***lattice**: Lattice Graphics*. R package version 0.20-24, URL http://lattice.R-Forge.R-project.org.

Sarkar D, Andrews F (2013). ***latticeExtra**: Extra Graphical Utilities Based on **lattice***. R package version 0.6-26, URL http://CRAN.R-project.org/package=latticeExtra.

Tableau Software (2011). *Software User Manual*. URL http://www.tableausoftware.com/support/manuals.

Zeileis A, Hornik K, Murrell P (2009). "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis*, **53**(9), 3259–3270.

Zeileis A, Meyer D, Hornik K (2007). "Residual-Based Shadings for Visualizing (Conditional) Independence." *Journal of Computational and Graphical Statistics*, **16**(3), 507–525.

Affiliation:

Richard M. Heiberger
Department of Statistics
Temple University
332 Speakman (006-12)
1810 N. 13 St.
Philadelphia, PA 19122-6083, United States of America
E-mail: rmh@temple.edu
URL: http://astro.ocis.temple.edu/~rmh/

Naomi B. Robbins
NBR
11 Christine Court
Wayne, New Jersey 07470-6523, United States of America
E-mail: Naomi@NBR-Graphs.com
URL: http://www.nbr-graphs.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 57, Issue 5 *Submitted:* 2011-10-19
March 2014 *Accepted:* 2013-07-24