Demonstration Document
Dotchartplus: Why and How

Jimmy Oh

Department of Statistics
University of Auckland

1 Introduction

Dotcharts are used to plot one quantitative variable with labels (Cleveland, 1985) and has many advantages over other ways of displaying labeled data. A flexible method for drawing dotcharts in R is presented in **A Literate Program for Drawing Dotcharts** (`dotchartplus.pdf`), which covers the technical details of the code. This is a *Demonstration Document* that discusses why you should use a dotchart, how `dotchartplus` differs from `dotchart` (built-in R function) and `dotplot` (from the 'lattice' R package), and gives some examples on how to use `dotchartplus`. The figures will often have accompanying code that will produce a similar plot. Only the important arguments are shown, so the actual result will vary slightly. Many of the examples used for demonstration intentionally replicate or mimic dotcharts found in Cleveland, W. S. (1985) **The Elements of Graphing Data**, to demonstrate that such figures can be drawn using `dotchartplus`.

Contents

1 Introduction **1**

2 Examples **2**
2.1 Dotchart vs Standard Graphs 2
2.1.1 Dotchart vs Barplot 2
2.1.2 Dotchart vs Population Pyramid 4
2.1.3 Dotchart vs Piechart 6
2.2 Dotchartplus vs Other Methods 8
2.2.1 Dotchartplus vs Dotchart 8
2.2.2 Dotchartplus vs Dotplot 10
2.3 Additional examples 14

3 Helpful Code **19**
3.1 Code Overview 19
3.2 Call Methods 19
3.3 Demonstration Functions 21
3.4 Included Datasets 27

2 Examples

2.1 Dotchart vs Standard Graphs

2.1.1 Dotchart vs Barplot

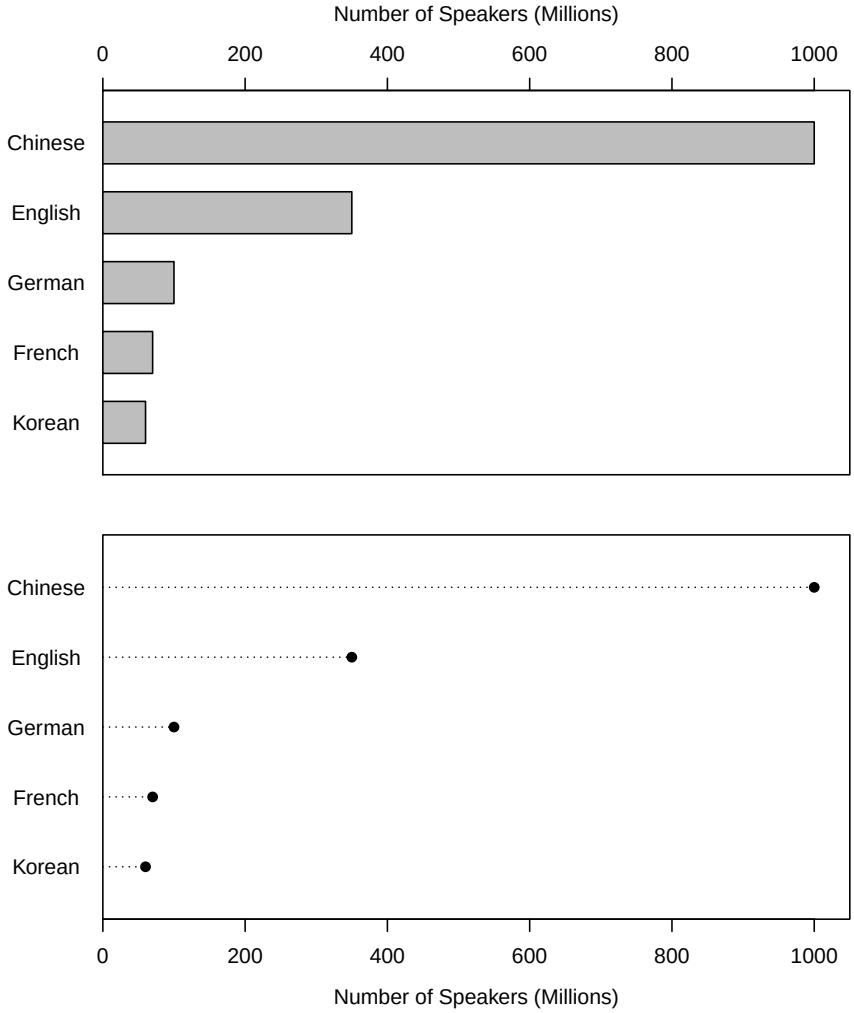
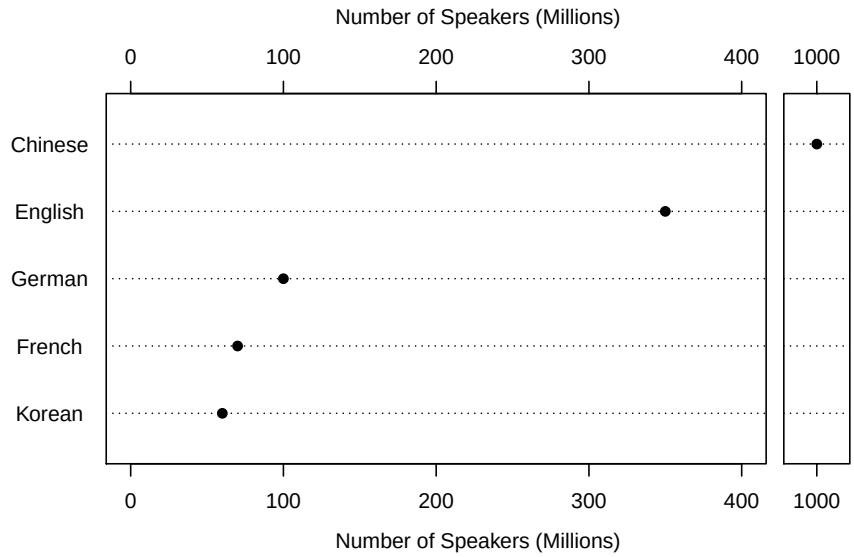


Figure 1: The numerical axis starts at 0 because the barplot is only meaningful when the length of the bars correspond to the data values. A dotchart can convey the same information, but with less ink. The call is:
`dotchartplus(rev(mother.tongue))`



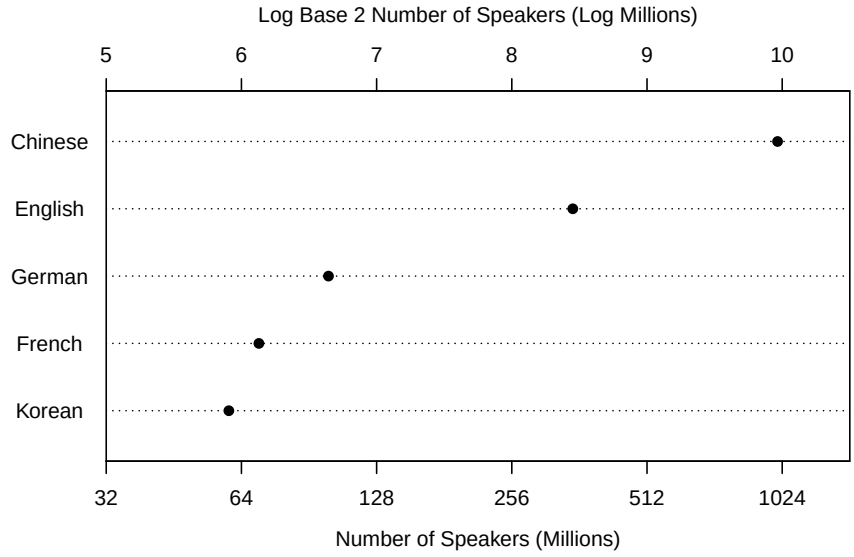


Figure 2: Unlike the barplot, a dotchart has flexibility on the numerical axis. This can include having breaks in the axis (top) or using a log scale (bottom). These types of plots are often inappropriate or misleading as barplots. The calls are:

```
dotchartplus(rev(mother.tongue), xlim = list(c(0, 400), c(980, 1020)))
dotchartplus(rev(log2(mother.tongue)))
```

2.1.2 Dotchart vs Population Pyramid

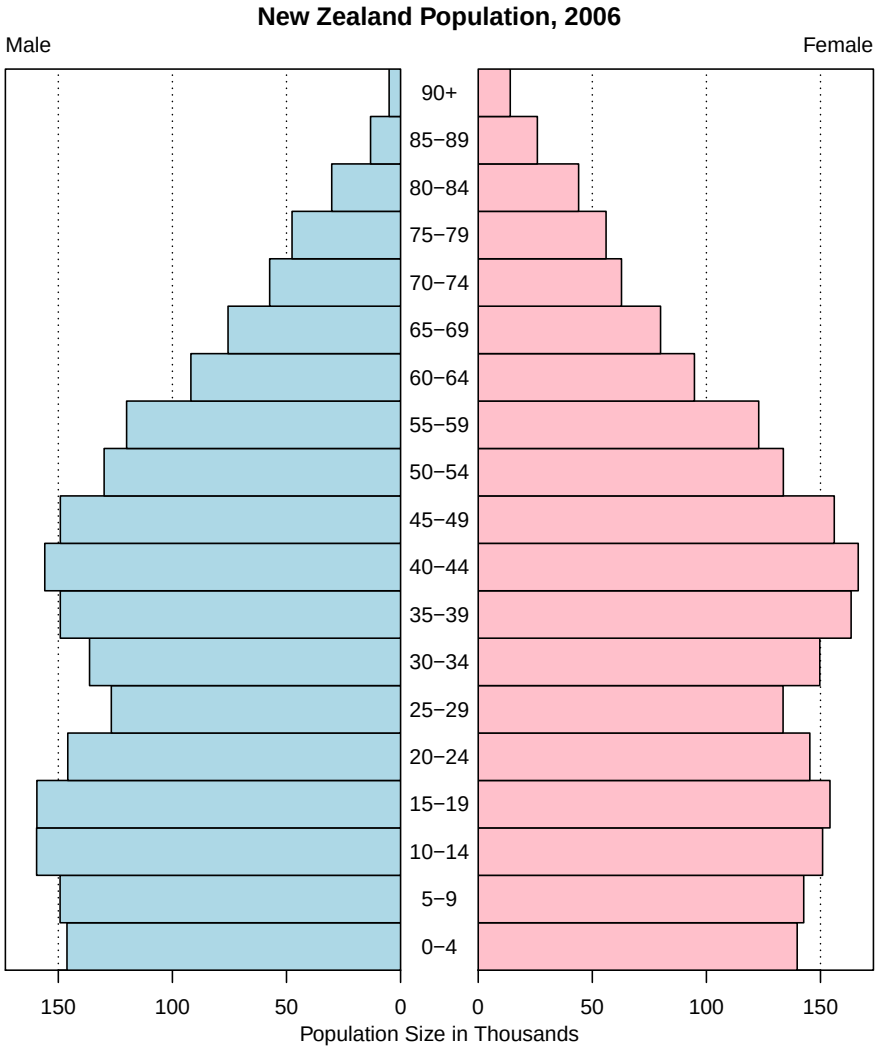


Figure 3: A population pyramid is often used to display the distribution of the population across age groups and between genders. In a growing population, the result looks like a pyramid as there will be less people as you move up the age groups. Unfortunately, the numerical axes for Male and Female go in opposite directions, which can make direct comparisons between the two difficult. The reader must frequently consult the axis labels.

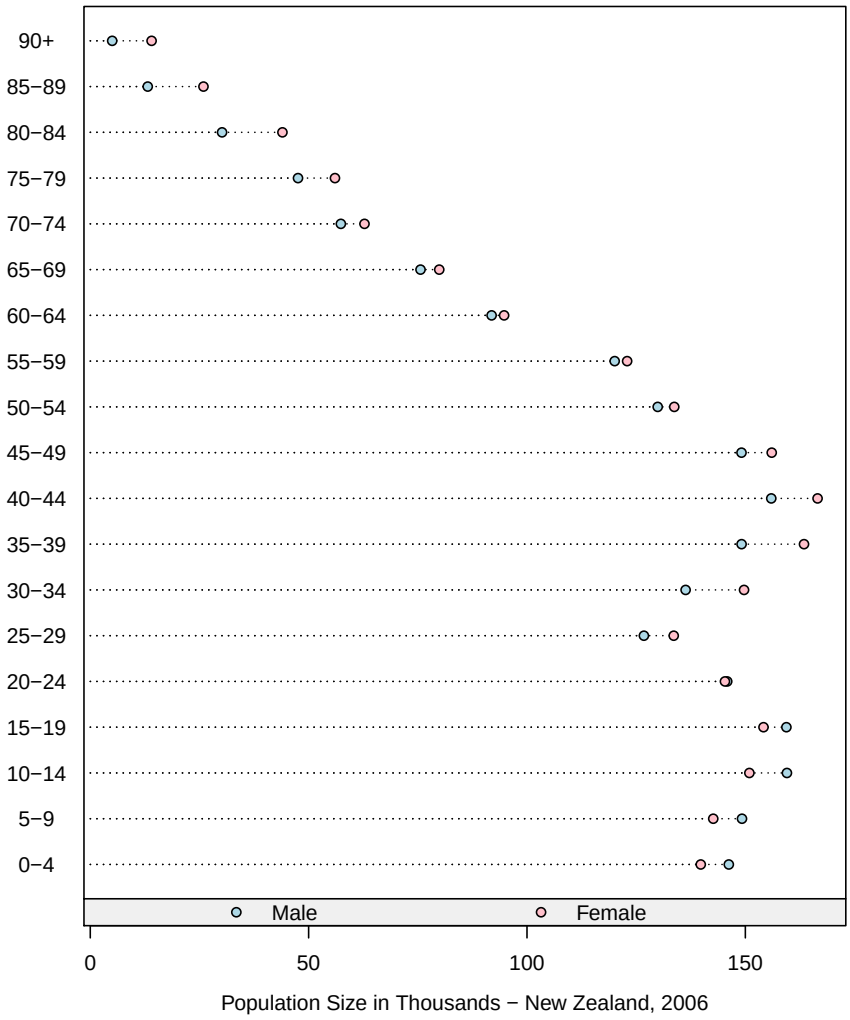


Figure 4: A dotchart provides an alternative to the population pyramid and has the advanage of being able to superpose multiple *sets* of data on a single plot. This makes direct comparisons between the two genders at each age group incredibly easy. The call is:
`dotchartplus(nzpoplist[[12]], col = c("lightblue", "pink"))`

2.1.3 Dotchart vs Piechart

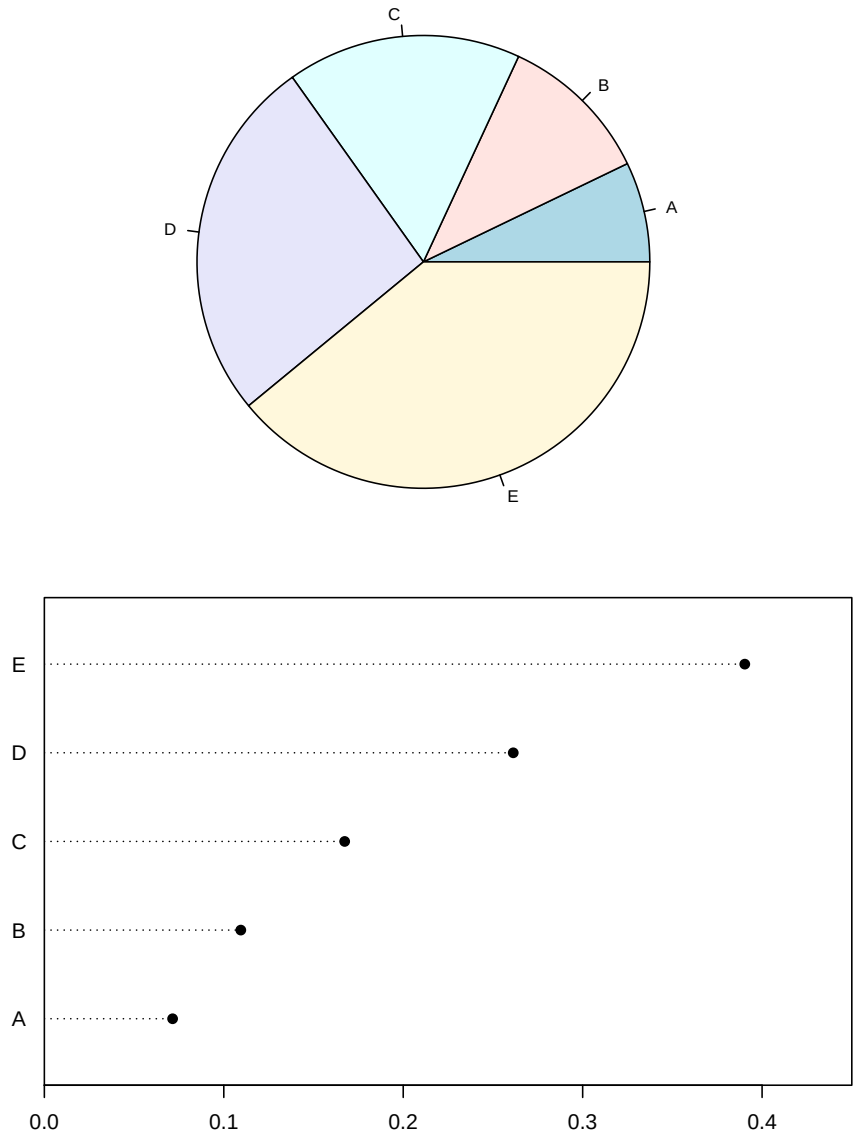


Figure 5: A piechart is a common tool for examining proportions. However, comparisons between slices require comparisons of angles or areas. These are far less accurate than judgements on lengths. This data happens to have the values sorted in ascending order, thus it is clear that group B has a larger proportion than group A even on the piechart.

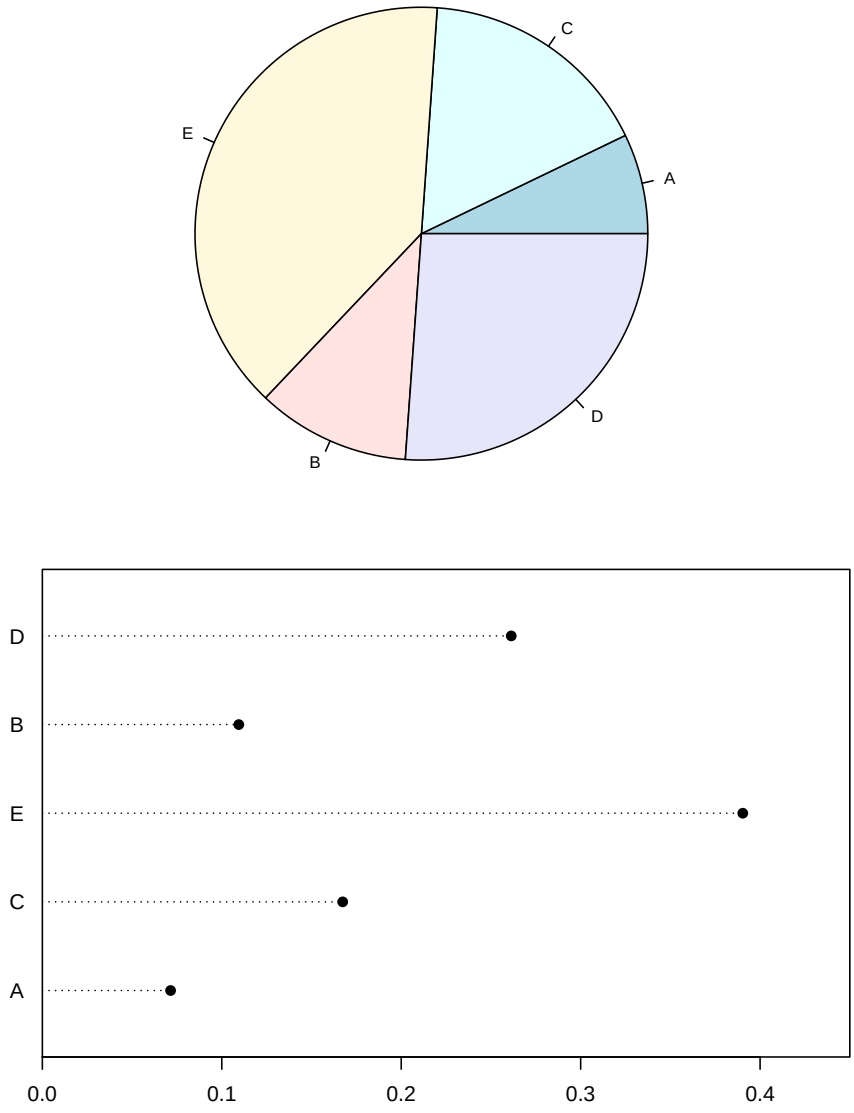


Figure 6: However, when the data is not sorted, it is not so easy to determine which is the larger proportion on the piechart. In contrast, the dotchart gives accurate readings of the actual proportions regardless of whether or not the data is sorted. It should also be noted that a piechart must be a perfect circle to be a proper piechart (not an ellipse). This is a cumbersome restriction that the dotchart does not have.

2.2 Dotchartplus vs Other Methods

2.2.1 Dotchartplus vs Dotchart

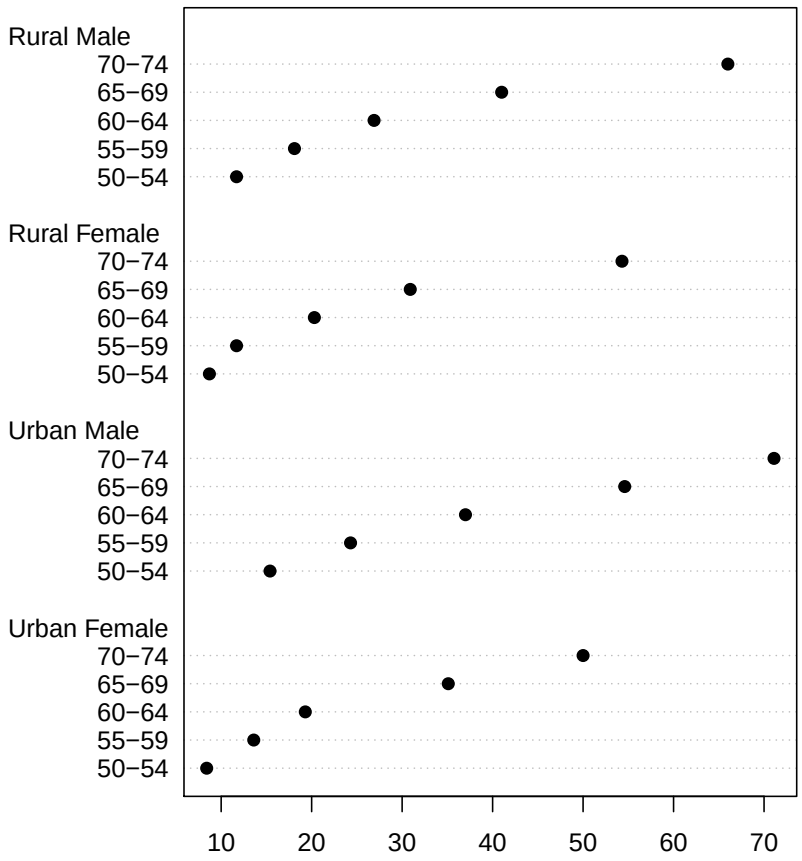


Figure 7: R's built-in `dotchart` function is a perfectly good function for plotting basic dotcharts. It also has the capacity to juxtapose *groups* of data. Refer to `help(dotchart)` for more information. The call is:
`dotchart(VADeaths)`

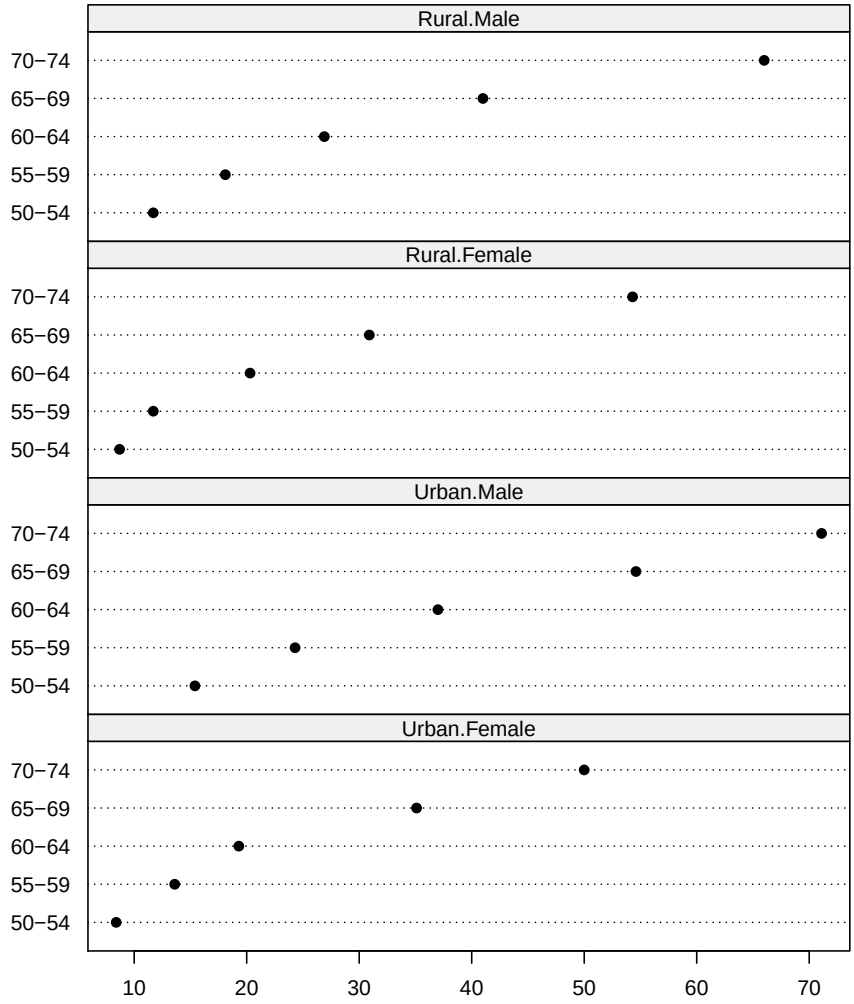


Figure 8: While there are some stylistic differences, `dotchartplus` is capable of doing everything `dotchart` can, and more. It's worth noting that the handling of `matrix` inputs differ between the two functions. In `dotchart`, columns of a `matrix` are taken to mean *groups* of data to juxtapose. In `dotchartplus`, columns are taken to mean *sets* of data to superpose. To juxtapose in `dotchartplus`, the data should be organised into a `list`, where each element of the `list` are taken to mean *groups* of data to juxtapose. Equivalently, a `data.frame` can be used, which has the advantage that the columns of a `data.frame` correspond to *groups* of data. Unfortunately, converting a `matrix` to a `data.frame` will strip the `row.names`, which are used as the text labels. The call is:

```
dotchartplus(data.frame(VADeaths), rep(list(row.names(VADeaths)), 4))
```

2.2.2 Dotchartplus vs Dotplot

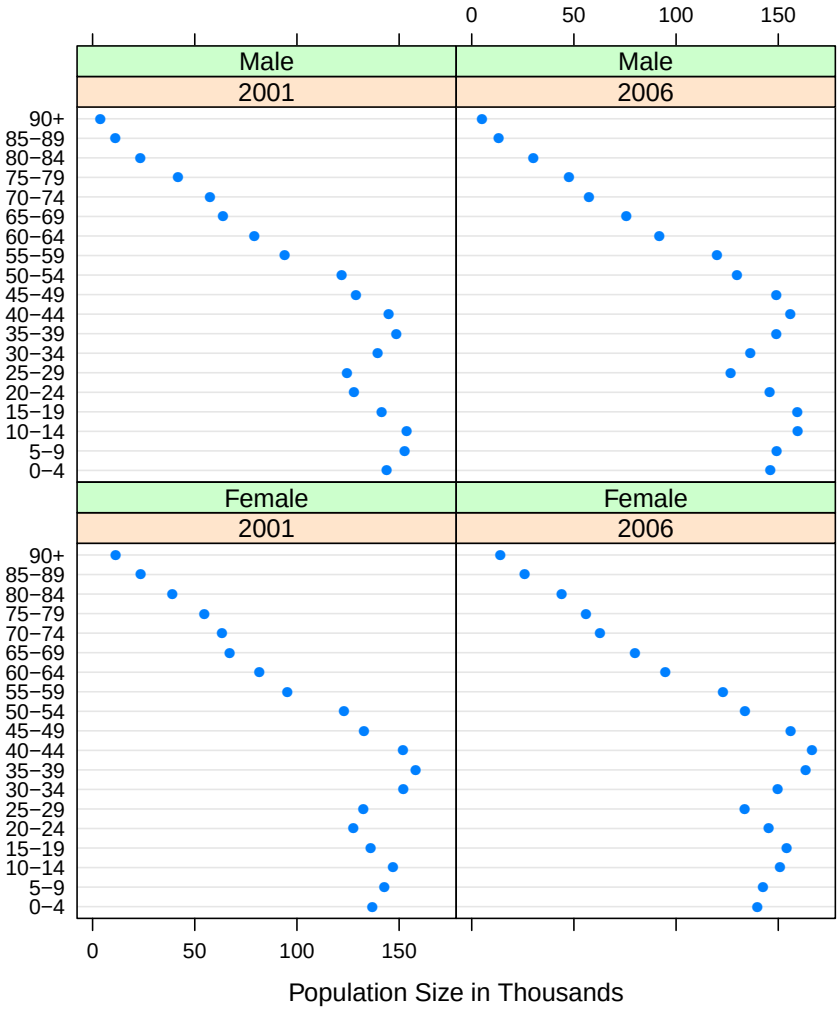


Figure 9: The `dotplot` function from the 'lattice' R package is a powerful function with many capabilities. In particular, it juxtaposes both horizontally and vertically, allowing for it to condition on many factors at the same time. It can also take more than one conditioning variable. Since `dotplot` does these things so well, there is no point in trying to compete with `dotchartplus`. The call is:

```
dotplot(group ~ freq | as.factor(year) * sex,
        data = nzpopdf[nzpopdf$year <= 2007 & nzpopdf$year >= 2000,],
        xlab = "Population Size in Thousands")
```

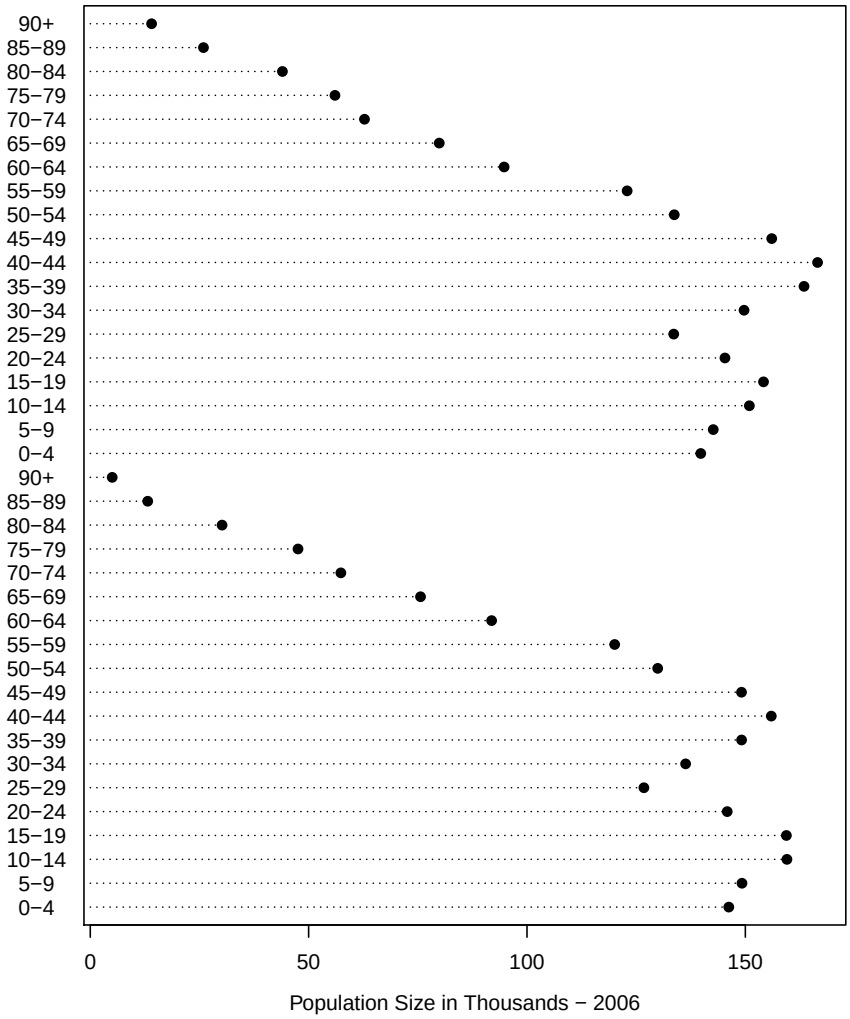


Figure 10: There is a note worthy difference in how `dotplot` and `dotchartplus` handle labels. First, `dotchartplus` takes its inputs very literally, so if the labels happen to be a `factor`, it will take these to be integers. The user must manually convert these to `character` by calling `as.character` if they want sensible labels. Further, this 'literal' processing means that even if one or more of the labels are the same, `dotchartplus` will simply plot them again. On the other hand, `dotplot` consolidates the labels and will superpose data with the same label (try it yourself using the same call as below, but using `dotplot`). The call is:

```
dotchartplus(as.character(group) ~ freq,
             data = nzpopdf[nzpopdf$year == 2006,],
             xlab = "Population Size in Thousands")
```

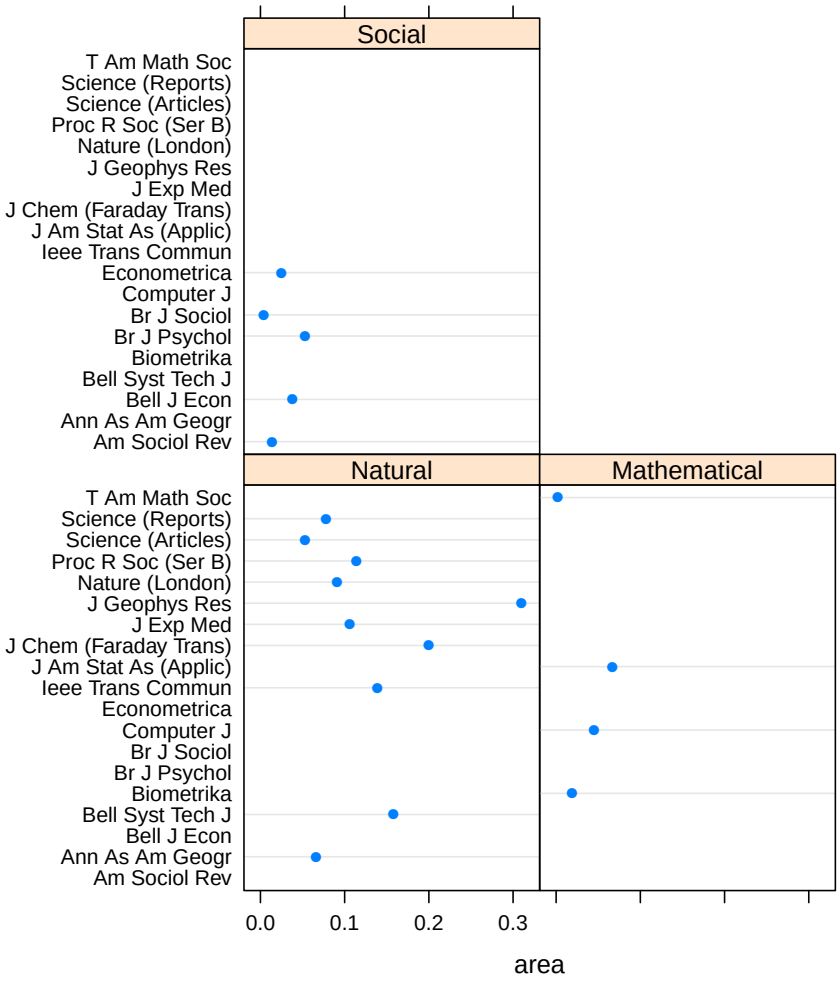


Figure 11: Because `dotplot` juxtaposes horizontally as well as vertically and automatically consolidates labels, it can draw some truly weird graphs as in this figure. `dotplot` expects an observation for each label at every level of the conditioning factor(s), but in this case the conditioning factor is a grouping variable and labels are not repeated. Further, by juxtaposing horizontally, it is harder to compare the values in 'Mathematical' to the other two groups. The call is:

```
dotplot(journal ~ area | science, data = jg)
```

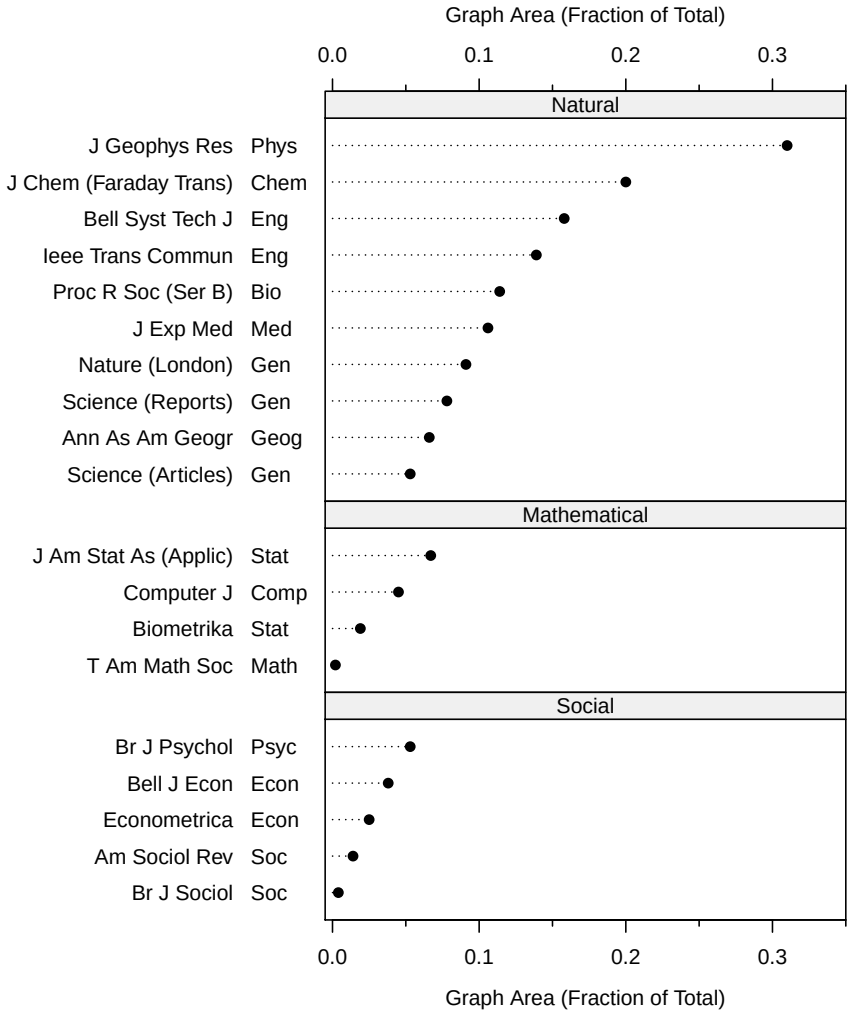


Figure 12: In comparison, `dotchartplus` only juxtaposes vertically and handles labels 'literally', making it ideal for plotting this kind of data. However, juxtaposing only vertically limits the number of levels the conditioning factor can have, for the simple reason that eventually the plot will become too squashed. In addition, `dotchartplus` can only handle one conditioning variable, whereas `dotplot` can handle many. Likewise, there are things that `dotchartplus` can do that `dotplot` can't, such as having multiple columns of text labels.
The moral of the story is, deciding whether to use `dotplot` or `dotchartplus` should depend on what kind of plot the user desires. The call is:
`dotchartplus(journal + subject ~ area | science, data = jg)`

2.3 Additional examples

By now, the reader should have a grasp on the calls required to plot dotcharts, either using `dotchart`, `dotchartplus` or `dotplot`. This subsection gives several more examples, showing off various features of `dotchartplus`.

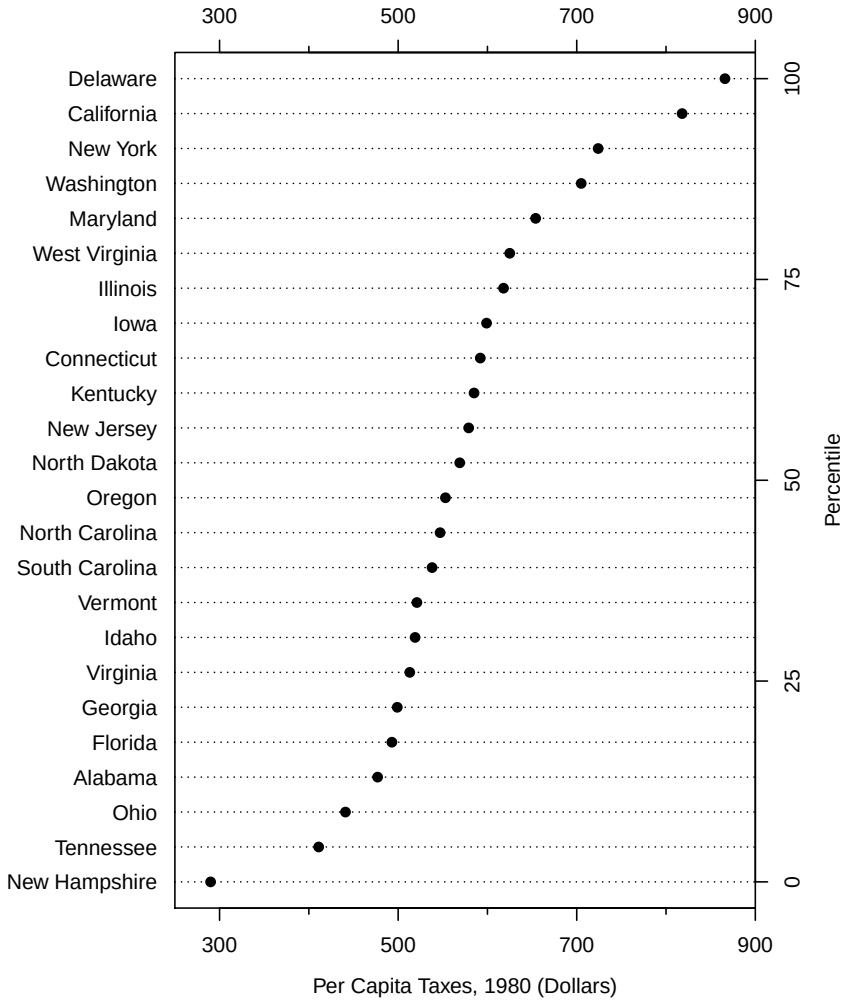


Figure 13: A plot designed to mimic the dotchart found in **The Elements of Graphing Data** (Cleveland, 1985, p147). The original data contains 48 states, but for the purposes of this document, only half the states (24) are used. "When the data are ordered from smallest to largest, the dotchart provides a percentile graph" (Cleveland, 1985, p147). The call is:
`dotchartplus(taxes, percentile = TRUE)`

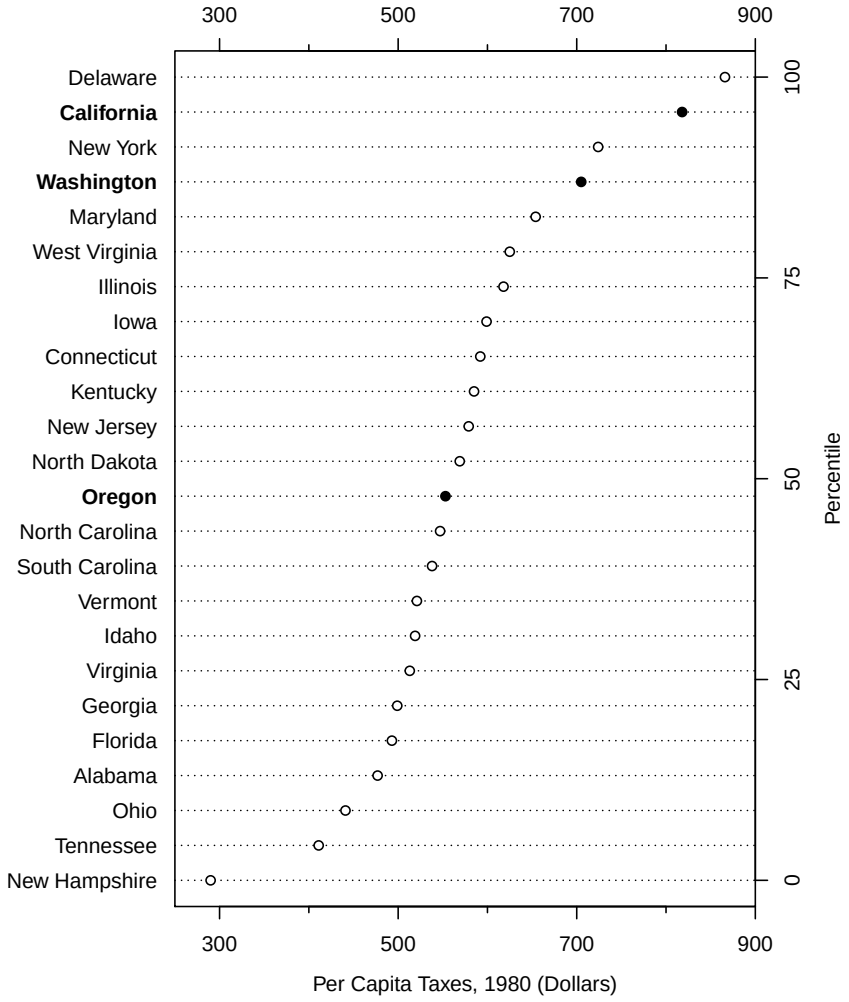


Figure 14: It is possible to highlight specific points easily in `dotchartplus` using the `highlight` argument. Here we choose to highlight the 3 states on the west coast of the US: Oregon (point 12), Washington (21) and California (23). The `highlight` argument can take a `numeric vector` specifying the indices of the points to highlight. The call is:

```
dotchartplus(taxes, highlight = c(12, 21, 23))
```

Note that because the dataset used here is a subset of the full data, the indices of the chosen states will not be the same. They will instead be: Oregon (23), Washington (41), California(46).

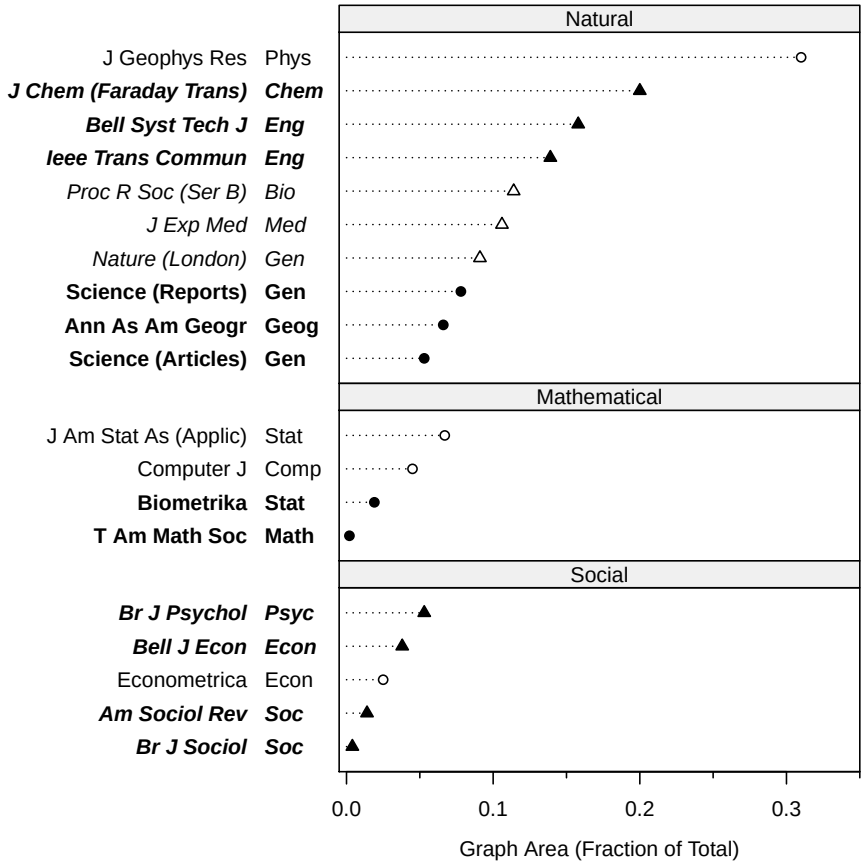


Figure 15: This figure is used to demonstrate how to make a more complex call using `highlight`. The call is:

```
highlight = list(matrix(1:9, nrow = 3),
                 c(1, 2),
                 matrix(c(0, 0, -3), ncol = 3)))
```

Each element of the `list` highlights the matching *group*. For each element of the `list`, one can specify a subsetting `numeric vector`, or multiple `numeric vectors` as columns of a `matrix`. Each column of the `matrix` will specify a distinct highlighting method (to the extent specified in `DefaultParslist`). As a `matrix` requires the same number of rows for each column, one can use `0` as a filler. As with subsetting in R, one can specify a negative index to signify every value except for that index. Full documentation on the usage of `highlight` can be found in the literate document for `dotchartplus`.

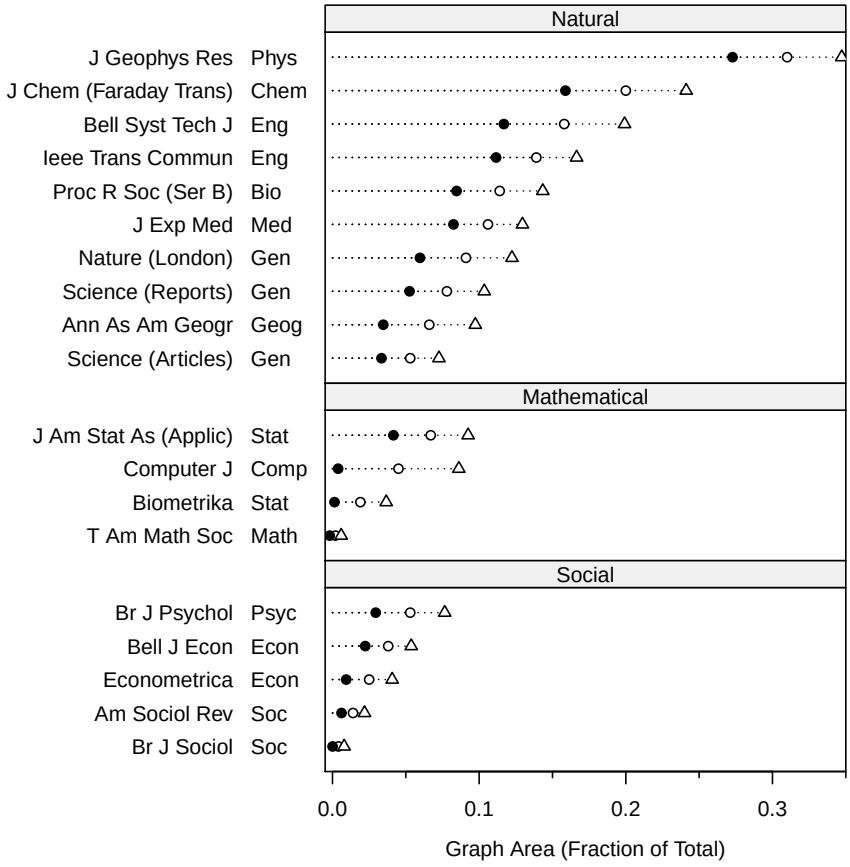


Figure 16: The data used in Figures 12 and 15 also contains a column called `stderr`. Naturally, this makes us want to plot confidence intervals. While a normality assumption may be questionable, we will use it anyway for demonstration purposes. We will do the popular 95% Confidence Interval by taking 1.96 times the standard error on either side. The `formula` method of calling `dotchartplus` allows specification of multiple *sets* of data by simply adding them in using `+`, so plotting the thing isn't too hard, but it certainly doesn't look very good. The call is:

```
dotchartplus(journal + subject ~ area + I(area - 1.96 * stderr) +
             I(area + 1.96 * stderr)| science, data = jg)
```

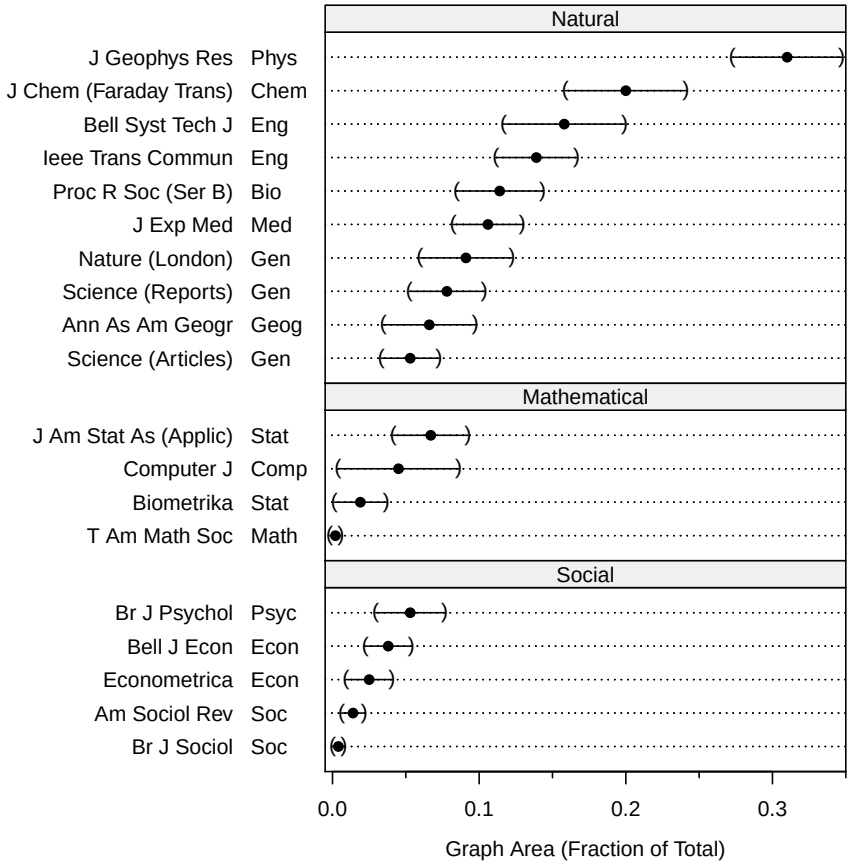


Figure 17: `dotchartplus` has a significant number of optional arguments that allow for extensive customisation of the output. The first thing we can change are the point types (`pch`) to something more fitting a confidence interval. The numbers 40 and 41 correspond to '(' and ')', which seem adequate. The other thing we can do is make the line inside the interval stand out more. This can be accomplished by specifying a custom points function (`pfunc`). One such function is defined in Subsection 3.3 called `pointsCI`, which will draw such a line for us. Our call now looks like this:

```
dotchartplus(journal + subject ~ area + I(area - 1.96 * stderr) +
             I(area + 1.96 * stderr)| science, data = jg,)
             pfunc = pointsCI, pch = c(21, 40, 41)
```

3 Helpful Code

On Literate Programs

This software is presented as a *literate program* written in the *noweb* format. It serves as both documentation and as a container for the code. A single `noweb` file can be used to both produce the *literate document* `pdf` file and to extract executable code. The document is separated into *documentation chunks* and named *code chunks*. Each *code chunk* can contain code or references to other *code chunks* which act as placeholders for the contents of the respective *code chunks*. As the name serves as a short description of the code, each *code chunk* can give an overview of what it does via the names it contains, leaving the reader free to delve deeper into the respective *code chunks* for the code if desired.

3.1 Code Overview

The default method covered in the literate document can be quite technical, so we will also be adding some easier interfaces (called *Call Methods*), including a `formula` interface that many R users should be familar with. This section also contains the *Demonstration Functions* and *Included Datasets*.

19a ⟨*dotchartplus-demos.R* 19a⟩≡
⟨*document header* 26b⟩
⟨*Call Methods* 19b⟩
⟨*Demonstration Functions* 21⟩
⟨*Included Datasets* 27⟩

This code is written to file `dotchartplus-demos.R`.

3.2 Call Methods

The default method can actually handle a variety of object types, the only requirement being that they are first put in a `list`.

Thus, we can add support for `matrix` and vectors of type `integer`, `numeric` and `logical`, simply by specifying a trivial function that places the object in a `list`, and then passing on all the arguments.

19b ⟨*Call Methods* 19b⟩≡

```
dcpsetmet = function(objectclass)
  setMethod("dotchartplus",
            signature(object = objectclass),
            function(object, ...) dotchartplus(list(object), ...)
            )
for(objclass in c("logical", "integer", "numeric", "matrix"))
  dcpsetmet(objclass)
rm(dcpsetmet)
```

⟨*define method for formula* 20⟩

Adding support for `formula` is quite a bit more complicated. The following code is partially based off the `formula` method for the `boxplot` function.

It uses the `parseFormula` function by Ross Ihaka to separate the formula into component parts: the data, labels and groups (see `formula.pdf` for more information). These are used to form the `list` format required by the default `dotchartplus` method. The formula can either be symbolic (names of variables in a `data.frame` specified in `data`) or the actual data itself, as is usual for other `formula` based calls in R. The `labels` and `group` can be specified either as part of the formula object, or separately as the arguments.

20 ⟨*define method for formula* 20⟩≡

```
setMethod("dotchartplus", signature(object = "formula"),
          function(object, labels = NULL, group = NULL,
                   data = NULL, ...)
 {
   m <- match.call(expand.dots = FALSE)
   ## Parse formula
   form.d = parseFormula(object)

   ## Adjust elements of m to required model.frame form
   m$formula = substitute(~ vals,
     list(vals = substitute(do.call("cbind", subs),
            list(subs = form.d$rhs))))
   m$object = NULL
   if(!is.null(form.d$lhs))
     m$labels = substitute(do.call("cbind", subs),
                  list(subs = form.d$lhs))
   if(!is.null(form.d$condition))
     m$group = form.d$condition[[1]]

   ## Convert to model.frame and evaluate
   m$... <- NULL
   m[[1L]] <- as.name("model.frame")
   mf <- eval(m, parent.frame())

   datcol = 1
   textcol = which(names(mf) == "(labels)")
   splitcol = which(names(mf) == "(group)")

   ## If group specified, split
   if(length(splitcol) != 0){
     datlist = split(data.frame(mf[,datcol]), mf[,splitcol])
     #datlist = split(mf[,datcol, drop = FALSE], mf[,splitcol])
     textlist = if(length(textcol) != 0)
       split(data.frame(mf[,textcol]), mf[,splitcol])
     else NULL
   } else{
     datlist = list(mf[,datcol])
     textlist = if(length(textcol) != 0)
       list(data.frame(mf[,textcol]))
```

```
      else NULL
    }

    dotchartplus(object = datlist, textlist = textlist, ...)
  }
)
```

3.3 Demonstration Functions

Certain Auxiliary Functions that may be useful are also included here. We also define a trivial function (`dotchartplus.runtests`) that runs each demo function in sequence in a single call.

21 ⟨*Demonstration Functions* 21⟩≡

⟨*Auxiliary Functions* 22⟩
⟨*mothertongue* 23⟩
⟨*journalgraphs* 24⟩
⟨*taxes* 25a⟩
⟨*radiation* 25b⟩
⟨*animalSpeed* 25c⟩
⟨*nzpop* 26a⟩

```
dotchartplus.runtests =
  function(...){
    demofuncs = ls(.GlobalEnv, pattern = "dotchartplus.demo")
    for(func in demofuncs){
      print(func)
      eval(parse(text = paste(func, "()", sep = "")))
      readline("Hit enter to run the next test.")
    }
  }
```

`lfuncBar` - a `lfunc` (line function) that draws a rectangle instead of dotted lines. This is used to plot a barplot using `dotchartplus` (see demo1.bar).

`idnfunc` - the I Do Nothing function. This is used to suppress actions. It can be used as the `pfunc` (point function) to suppress points when drawing barplots (see demo1.bar). It can also be used as the `lfunc` to suppress lines, although that can also be accomplished by setting `lty = 0`.

`pointsCI` - a `pfunc` (points function) that draws Confidence Intervals. Refer to Figure 17.

`dcpTranspose` - A method of 'transposing' or flipping a `datlist`, switching what is juxtaposed and what is superposed (compare demo6.yearBYgender and demo6.genderBYyear).

22 ⟨*Auxiliary Functions* 22⟩≡

```
lfuncBar = function(xl, yb, xr, yt, ...)
  rect(xl, yb - 0.3, xr, yt + 0.3, ...)

idnfunc = function(...)
  NULL

pointsCI = function(x, y, pch, ...){
  segments(x[pch == 40], y[pch == 21], x[pch == 41], y[pch ==21])
  points(x, y, pch = pch, ...)
}

dcpTranspose = function(datlist){
  datlist = lapply(datlist, as.matrix)
  newlist = list()
  for(i in 1:ncol(datlist[[1]]))
    newlist[[i]] = sapply(datlist, function(x) x[,i])
  names(newlist) = dimnames(datlist[[1]])[[2]]
  newlist
}
```

Demonstration functions using the `mother.tongue` data.

23 ⟨*mothertongue* 23⟩≡

```
dotchartplus.demo1 =
  function(...)
  dotchartplus(rev(mother.tongue), axes = 1:3,
               xlim = c(0, 1000),
               xlab = "Number of Speakers (Millions)", ...)

dotchartplus.demo1.prop =
  function(...)
  dotchartplus(rev(mother.tongue[-1]/max(mother.tongue)),
               axes = 1:3, xlim = c(0, 0.35),
               lab1 = paste("Number of Language Speakers,",
                 "proportional to\nThe Number of Chinese",
                 "Speakers (1 Billion)"), ...)

dotchartplus.demo1.xlim =
  function(axes = 1:3, ...)
  dotchartplus(rev(mother.tongue), axes = axes,
               xlim = list(c(0, 400), c(980, 1020)),
               at = list(NULL, 1000),
               xlab = "Number of Speakers (Millions)",
               ...)

dotchartplus.demo1.log =
  function(axes = 1:3, ...){
    mt.log = log2(mother.tongue)
    dotchartplus(rev(mt.log), axes = axes, xlim = c(5, 10.5),
                 xaxs = "i", at = 5:11, atlabels1 = 2^(5:11),
                 lab1 = "Number of Speakers (Millions)",
                 lab3 = paste("Log Base 2 Number of",
                   "Speakers (Log Millions)"), ...)
  }

dotchartplus.demo1.bar =
  function(lcol = "grey", lty = 1, ...)
  dotchartplus(rev(mother.tongue), axes = 1:3,
               xlim = c(0, 1000),
               xlab = "Number of Speakers (Millions)",
               lfunc = lfuncBar, lcol = lcol, lty = lty,
               pfunc = idnfunc, ...)
```

Demonstration functions using the `jg` data.

Two functions are provided to plot the same figure, one using the default `dotchartplus list` method, the other using the `formula` method.

24 ⟨*journalgraphs* 24⟩≡

```
dotchartplus.demo2 =
  function(axes = 1:3, ...){
    ## factor groups into Natural, Mathematical, Social
    jg$science = factor(jg$science,
      levels = c("Natural", "Mathematical", "Social"))

    ## sort by area
    jg.sort = jg[order(jg$area),]

    ## separate into textlist and datlist
    ## split into science groups
    jg.textlist = split(jg.sort[,1:2], jg.sort$science)
    jg.datlist = split(jg.sort[,4], jg.sort$science)

    dotchartplus(jg.datlist, jg.textlist, axes = axes,
                 xlim = c(-.005, 0.35), xaxs = "i",
                 xlab = "Graph Area (Fraction of Total)",
                 adj = c(1, 0), at = (0:3)/10,
                 atsmall = (0:3)/10 + 0.05, ...)
  }

dotchartplus.demo2.formula =
  function(...){
    ## factor groups into Natural, Mathematical, Social
    jg$science = factor(jg$science,
      levels = c("Natural", "Mathematical", "Social"))

    dotchartplus(journal + subject ~ area | science,
                 data = jg, axes = 1:3,
                 xlim = c(-.005, 0.35), xaxs = "i",
                 xlab = "Graph Area (Fraction of Total)",
                 adj = c(1, 0), at = (0:3)/10,
                 atsmall = (0:3)/10 + 0.05, ...)
  }

dotchartplus.demo2.CI =
  function(...){
    dotchartplus(journal + subject ~ area + I(area - 1.96 * stderr) +
                 I(area + 1.96 * stderr)| science, data = jg,
                 setslabel = FALSE, axes = 1:2,
                 xlim = c(-.005, 0.35), xaxs = "i",
                 xlab = "Graph Area (Fraction of Total)",
                 adj = c(1, 0), at = (0:3)/10,
                 atsmall = (0:3)/10 + 0.05, full.lines = TRUE,
                 pfunc = pointsCI, pch = c(21, 40, 41))
```

```
}
```

Demonstration function using the `taxes` data.

25a ⟨*taxes* 25a⟩≡

```
dotchartplus.demo3 =
  function(...)
  dotchartplus(taxes, axes = 1:3, xlim = c(250, 900),
               xaxs = "i", adj = 1, percentile = TRUE,
               lab1 = "Per Capita Taxes, 1980 (Dollars)",
               atsmall = seq(400, 800, length = 3),
               at = seq(300, 900, length = 4), ...)
```

Demonstration function using the `radiation` data.

25b ⟨*radiation* 25b⟩≡

```
dotchartplus.demo4 =
  function(...)
  dotchartplus(radiation, lty = 0, axes = 1:3,
    lab1 = "Log Base 10 (Extragalactic / Galactic)", ...)
```

Demonstration functions using the `animalSpeed` data.
The data is in miles, but we convert it to km to match the figure in Cleveland.

25c ⟨*animalSpeed* 25c⟩≡

```
dotchartplus.demo5 =
  function(...)
  dotchartplus(rev(animalSpeed * 1.6093), adj = 1,
               axes = 1:3, lab1 = "Speed (km/hr)", ...)

dotchartplus.demo5.xlim =
  function(...)
  dotchartplus(rev(animalSpeed * 1.6093),
               xlim = list(c(0, 2), c(12.5, 112.5)),
               at = list(c(0, 2), seq(25, 100, by = 25)),
               atsmall = list(1,
                 seq(12.5, 112.5, by = 12.5)),
               widths = c(1, 4), adj = 1, axes = 1:3,
               lab1 = "Speed (km/hr)", ...)
```

Demonstration functions using the nzpop data.

26a ⟨*nzpop* 26a⟩≡

```
dotchartplus.demo6 =
  function(subs = c("1951", "2006"), ...){
    nzpopsubmat = sapply(nzpoplist[subs],
      function(x) apply(x, 1, sum))
    xlab = "New Zealand Population Size in Thousands"
    dotchartplus(nzpopsubmat, xlab = xlab, ...)
  }

dotchartplus.demo6.scaled =
  function(subs = c("1951", "2006"), ...){
    nzpopsubmat = sapply(nzpoplist[subs],
      function(x) apply(x, 1, sum))
    nzpopsubmat.scaled = sweep(nzpopsubmat, 2,
      apply(nzpopsubmat, 2, sum), "/")
    xlab =
      "Proportion of New Zealand Population per Age Group"
    dotchartplus(nzpopsubmat.scaled, xlab = xlab, ...)
  }

dotchartplus.demo6.babies =
  function(...){
    babies = dcpTranspose(lapply(nzpoplist, function(x)
      apply(x, 1, sum)))[[1]][1,][1:12]
    xlab = paste("New Zealand Population in Thousands",
      "between 0-4 years old")
    dotchartplus(babies, xlab = xlab, ...)
  }

dotchartplus.demo6.yearBYgender =
  function(subs = c("1951", "2006"), ...){
    xlab = "New Zealand Population Size in Thousands"
    dotchartplus(dcpTranspose(nzpoplist[subs]), xlab = xlab,
                 ...)
  }

dotchartplus.demo6.genderBYyear =
  function(subs = c("1951", "2006"), ...){
    xlab = "New Zealand Population Size in Thousands"
    dotchartplus(nzpoplist[subs], xlab = xlab, ...)
  }
```

We place a document header at the top of the extracted code to encourage people to read the literate description.

26b ⟨*document header* 26b⟩≡

```
## The code in this .R file is machine generated.
## To understand the program, read the literate description
##  pdf rather than studying just the R code.
```

3.4 Included Datasets

With the exception of the NZ Census data and the Language Speakers data, these are the same data used in Cleveland, W. S. (1985) **The Elements of Graphing Data**. The Language Speakers data is similar to the one used in the book, but not exactly the same. Sources are given where available, others were given to me from Ross Ihaka without a specified source.

27 ⟨*Included Datasets* 27⟩≡

```
##---------------------------------------
## Datasets (from Ross)
##---------------------------------------

## Per Capita Taxes
## From Cleveland's "The Elements of Graphing Data",
## First Edition. Page 147.
## Data captured by scanning and the use of "g3data".
## I have deliberately left the values unrounded.

taxes =
    round(c(290.016639308, 392.12318715, 410.645447819, 425.771230625,
            440.896757637, 474.890185047, 476.997116578, 490.991081152,
            493.098524269, 498.978268502, 499.198921578, 508.287232049,
            513.035925429, 515.897572976, 519.136578531, 519.922757035,
            521.086293546, 537.155215579, 538.319007883, 538.161535574,
            547.060911248, 549.922303003, 552.972885347, 554.136933443,
            568.884846656, 569.671280952, 578.758824045, 581.809406389,
            585.425514159, 586.777729675, 591.715102058, 596.463539645,
            599.324675607, 609.733739106, 618.444435776, 619.607972287,
            624.922191094, 642.123698735, 654.419552273, 684.639143811,
            705.425552528, 711.683422147, 714.733748698, 723.821291791,
            785.739723373, 817.658193324, 823.538449142, 866.400045531))
names(taxes) =
  c("New Hampshire", "South Dakota", "Tennessee", "Missouri", "Ohio",
    "Texas", "Alabama", "Indiana", "Florida", "Mississippi", "Georgia",
    "Arkansas", "Virginia", "Colorado", "Idaho", "Nebraska", "Vermont",
    "Kansas", "South Carolina", "Utah", "North Carolina", "Maine",
    "Oregon", "Montana", "North Dakota", "Louisiana", "New Jersey",
    "Rhode Island", "Kentucky", "Oklahoma", "Connecticut", "Nevada",
    "Iowa", "Pennsylvania", "Illinois", "Arizona", "West Virginia",
    "Michigan", "Maryland", "Massachusetts", "Washington", "New Mexico",
    "Wisconsin", "New York", "Minnesota", "California", "Wyoming",
    "Delaware")

# Source: The Cambridge Factfinder, Cambridge University Press, 1993
mother.tongue =
structure(
          c(1000, 350, 250, 200, 150, 150, 150, 135, 120,
            100, 70, 70, 65, 65, 60, 60, 55, 55, 50, 50),
          names =
          c("Chinese", "English", "Spanish", "Hindi", "Arabic",
            "Bengali", "Russia", "Portuguese", "Japanese", "German",
```

```
            "French", "Panjabi", "Javanese", "Bihari", "Italian",
            "Korean", "Telugu", "Tamil", "Marathi", "Vietnamese"))

## Page 146, Elements, 1st Ed
## Radiation Sources

radiation = structure(c(-0.5, 1.0, -2.5, -1.5, -2.0, 3.5, -1.0),
                      .Names = c("Gamma Rays", "X-Rays",
                                 "Ultraviolet\nRadiation",
                                 "Visible Light",
                                 "Infrared\nRadiation",
                                 "Microwaves", "Radio Emission"))

animalSpeed =
  structure(c(70, 61, 50, 50, 50, 47.5, 45, 45, 43, 42, 40, 40,
              40, 39.35, 35.5, 35, 35, 35, 32, 32, 30, 30, 30,
              30, 27.89, 25, 20, 18, 15, 12, 11, 9, 1.17, 0.17,
              0.15, 0.03),
            .Names = c("Cheetah", "Pronghorn Antelope", "Wildebeest",
                       "Lion", "Thomson's Gazelle", "Quarterhorse",
                       "Elk", "Cape Hunting Dog", "Coyote", "Gray Fox",
                       "Hyena", "Zebra", "Mongolian Wild Ass",
                       "Greyhound", "Whippet", "Rabbit (domestic)",
                       "Mule Deer", "Jackal", "Reindeer", "Giraffe",
                       "White-Tailed Deer", "Wart Hog", "Grizzly Bear",
                       "Cat (domestic)", "Human", "Elephant",
                       "Black Mamba Snake", "Six-Lined Race Runner",
                       "Wild Turkey", "Squirrel", "Pig (domestic)",
                       "Chicken", "Spider (Tegenaria atrica)",
                       "Giant Tortoise", "Three-Toed Sloth",
                       "Garden Snail"))

jg =
data.frame(
           journal =
           c("J Soc Psychol", "Oxford Rev Educ", "Br J Sociol",
             "Rev Educ Res", "Social Forces", "Am Sociol Rev",
             "Am J Sociol", "Rev Econ Stat", "Econometrica",
             "Educ Res", "J Polit Econ", "Bell J Econ",
             "Am Econ Rev", "Am Educ Res", "Br J Psychol",
             "J Exp Psychol", "Percept Psychophys",
             "T Am Math Soc", "Ann Statist", "Computer",
             "Biometrika", "Commun Acm", "Am Math Mo",
             "Computer J", "J Am Stat As (Theory)",
             "Comput Gra Im Proc", "J Am Stat As (Applic)",
             "J R Stat Soc (Ser C)", "Cell", "Science (Articles)",
             "Geogr Rev", "Geogr J", "Ann As Am Geogr",
             "New Eng J Med", "Ibm J Res Develop",
             "Science (Reports)", "Lancet", "Profess Geogr",
             "Nature (London)", "Phys Rev (A)", "J Am Chem Soc",
```

```
  "J Exp Med", "Phys Rev (Lett)", "Scientific Am",
  "Proc R Soc (Ser B)", "J Physics (C)",
  "J Clin Investigation", "Ieee Trans Commun",
  "J Phys Chem", "Life Sciences", "Bell Syst Tech J",
  "Ieee Comm Radar Sig", "J Appl Physics",
  "J Chem (Faraday Trans)", "J Exp Biol",
  "J Appl Polym Sci", "J Geophys Res"),
subject =
c("Psyc", "Educ", "Soc", "Educ", "Soc", "Soc", "Soc",
  "Econ", "Econ", "Educ", "Econ", "Econ", "Econ", "Educ",
  "Psyc", "Psyc", "Psyc", "Math", "Math", "Comp", "Stat",
  "Comp", "Math", "Comp", "Stat", "Comp", "Stat", "Stat",
  "Bio", "Gen", "Geog", "Geog", "Geog", "Med", "Eng",
  "Gen", "Med", "Geog", "Gen", "Phys", "Chem", "Med",
  "Phys", "Gen", "Bio", "Phys", "Med", "Eng", "Chem",
  "Bio", "Eng", "Eng", "Phys", "Chem", "Bio", "Chem",
  "Phys"),
science =
c("Social", "Social", "Social", "Social", "Social", "Social",
  "Social", "Social", "Social", "Social", "Social", "Social",
  "Social", "Social", "Social", "Social", "Social",
  "Mathematical", "Mathematical", "Mathematical",
  "Mathematical", "Mathematical", "Mathematical",
  "Mathematical", "Mathematical", "Mathematical",
  "Mathematical", "Mathematical", "Natural", "Natural",
  "Natural", "Natural", "Natural", "Natural", "Natural",
  "Natural", "Natural", "Natural", "Natural", "Natural",
  "Natural", "Natural", "Natural", "Natural", "Natural",
  "Natural", "Natural", "Natural", "Natural", "Natural",
  "Natural", "Natural", "Natural", "Natural", "Natural",
  "Natural", "Natural"),
area =
c(0.000, 0.001, 0.004, 0.004, 0.009, 0.014, 0.014, 0.014,
  0.025, 0.032, 0.034, 0.038, 0.038, 0.041, 0.053, 0.082,
  0.089, 0.002, 0.006, 0.014, 0.019, 0.023, 0.034, 0.045,
  0.057, 0.059, 0.067, 0.105, 0.034, 0.053, 0.055, 0.063,
  0.066, 0.068, 0.070, 0.078, 0.084, 0.089, 0.091, 0.097,
  0.098, 0.106, 0.108, 0.108, 0.114, 0.122, 0.127, 0.139,
  0.149, 0.157, 0.158, 0.164, 0.199, 0.200, 0.207, 0.264,
  0.31),
stderr =
c(0.000, 0.001, 0.002, 0.002, 0.005, 0.004, 0.005, 0.004,
  0.008, 0.012, 0.008, 0.008, 0.007, 0.011, 0.012, 0.020,
  0.014, 0.002, 0.003, 0.005, 0.009, 0.008, 0.010, 0.021,
  0.020, 0.018, 0.013, 0.020, 0.007, 0.010, 0.015, 0.012,
  0.016, 0.009, 0.012, 0.013, 0.014, 0.018, 0.016, 0.014,
  0.014, 0.012, 0.010, 0.019, 0.015, 0.016, 0.012, 0.014,
  0.017, 0.015, 0.021, 0.020, 0.013, 0.021, 0.019, 0.017,
  0.019),
stringsAsFactors = FALSE)
```

```
## Statistics NZ Census Data
nzpop =
  array(c(121, 96.72, 82.2, 67.98, 70.96, 74.62, 68.87, 69.12,
    66.42, 58.83, 49.52, 40.04, 37.79, 34.14, 26, 15.29, 7.07, 2.4,
    0.64, 133.09, 124.16, 98.97, 83.32, 70.64, 78.49, 79.67, 72.12,
    70.79, 66.98, 57.5, 47.5, 36.65, 32.78, 26.98, 18.5, 8.94, 3.27,
    0.9, 152.17, 136.14, 126.41, 99.74, 85.27, 74.59, 82.1, 81.87,
    73.34, 71.4, 65.35, 54.59, 43.08, 31.83, 25.8, 18.92, 10.41,
    3.79, 1.04, 155.37, 156.21, 138.87, 126.68, 100.83, 88.6, 76.69,
    83.79, 82.69, 72.94, 69.29, 62.15, 49.96, 37.86, 24.47, 17.75,
    10.39, 4.58, 1.18, 153.65, 156.47, 157.24, 136.03, 120.15, 98.6,
    87.95, 76.57, 82.81, 80.66, 70.44, 65.22, 56.34, 43.51, 29.44,
    16.85, 9.71, 4.45, 1.43, 149.59, 160.13, 163.25, 154.79, 130.63,
    123.92, 101.86, 89.99, 77.73, 82.53, 77.46, 65.41, 59.28, 48.46,
    33.32, 19.81, 9.88, 4.02, 1.41, 127.48, 146.2, 155.99, 155.76,
    136.54, 118.98, 118.18, 96.67, 85.26, 74.54, 78.05, 72.26, 58.52,
    50.86, 37.83, 23.35, 11.36, 4.15, 1.65, 127.82, 129.16, 147.48,
    153.34, 140.8, 133.11, 121.83, 119.33, 96.93, 83.6, 71.97, 74.49,
    66.47, 51.38, 41.28, 27.42, 14.03, 5.58, 1.84, 147.1, 133.28,
    132.47, 144.05, 143.18, 139.29, 139.74, 126.09, 122, 97.04, 84,
    72.11, 73.37, 62.45, 44.81, 31.9, 17.28, 6.96, 2.15, 152.25,
    151.79, 138.14, 137.9, 140.17, 139.25, 147.79, 145.57, 130.62,
    124.95, 96.75, 81.85, 68.35, 66.31, 52.79, 34.22, 20.83, 8.61,
    2.89, 144, 152.7, 153.8, 141.5, 128, 124.6, 139.6, 148.6, 145,
    128.9, 122, 94, 79.2, 63.8, 57.5, 41.8, 23.4, 11.1, 3.8, 146.21,
    149.23, 159.52, 159.4, 145.85, 126.77, 136.32, 149.14, 155.94,
    149.12, 129.93, 120.08, 91.9, 75.62, 57.38, 47.57, 30.18, 13.16,
    5.02, 161.8, 148.15, 150.99, 163.5, 165.93, 147.16, 133.73, 139.98,
    150.79, 155.13, 146.14, 126.44, 116.26, 87.85, 69.62, 49.2, 36.13,
    18.28, 6.87, 158.24, 163.17, 149.61, 154.11, 168.8, 165.09, 152.12,
    136.72, 141, 149.65, 151.89, 142.29, 122.68, 111.14, 81.4, 60.63,
    38.39, 22.72, 9.77, 157.2, 159.53, 164.55, 152.66, 159.27, 167.73,
    169.6, 154.84, 137.69, 139.93, 146.55, 148.06, 138.32, 117.74,
    103.43, 71.55, 48.24, 24.81, 13.02, 157.63, 158.51, 160.92, 167.59,
    157.87, 158.3, 172.31, 172.32, 155.77, 136.74, 137.06, 143.05,
    144.21, 133.12, 110.22, 91.62, 57.7, 32.18, 15.44, 157.8, 158.96,
    159.91, 163.99, 172.79, 156.96, 162.96, 175.07, 173.24, 154.78,
    134.04, 133.89, 139.64, 139.14, 125.14, 98.51, 74.74, 39.3, 20.44,
    157.41, 159.15, 160.37, 163, 169.23, 171.88, 161.68, 165.82,
    176.06, 172.22, 151.98, 131.1, 130.94, 135.15, 131.24, 112.6,
    81.46, 51.84, 26.37, 159.37, 158.77, 160.56, 163.48, 168.28,
    168.38, 176.6, 164.59, 166.92, 175.1, 169.36, 148.89, 128.45,
    127.04, 127.99, 118.7, 94.12, 57.82, 35.55, 161.86, 160.74, 160.19,
    163.68, 168.79, 167.47, 173.16, 179.52, 165.76, 166.1, 172.31,
    166.17, 146.02, 124.95, 120.69, 116.49, 100.04, 68.09, 42.97,
    162.22, 163.24, 162.17, 163.32, 169.02, 168.01, 172.29, 176.14,
    180.7, 165.02, 163.5, 169.21, 163.13, 142.17, 119.13, 110.32,
    99.18, 73.39, 52.24, 162.02, 163.61, 164.68, 165.31, 168.69,
    168.28, 172.87, 175.31, 177.38, 179.95, 162.53, 160.67, 166.28,
```

159, 135.76, 109.52, 94.6, 74.12, 59.6, 162.34, 163.41, 165.04,
167.83, 170.7, 167.98, 173.17, 175.93, 176.61, 176.71, 177.42,
159.85, 158.1, 162.28, 152.13, 125.24, 94.84, 71.59, 63.9, 115.98,
92.66, 79.48, 65.26, 68.43, 72.15, 70.25, 69.5, 63.9, 55.83,
49.1, 43.59, 40.48, 36.63, 28.09, 17.12, 8.46, 3.19, 1.03, 127.88,
118.58, 94.48, 80.65, 67.74, 72.88, 75.4, 73.13, 70.63, 64.44,
54.75, 47.88, 41.74, 37.96, 31.06, 21.88, 11.06, 4.52, 1.42,
148.32, 130.03, 120.64, 94.94, 81.51, 71.46, 75.44, 77.12, 73.74,
70.37, 63.43, 53.23, 45.82, 38.7, 33.2, 24.1, 14.39, 5.15, 1.75,
148.21, 149.99, 132.46, 120.56, 97.23, 85.53, 73.57, 78.19, 79.12,
73.97, 70.46, 61.5, 51.29, 43.47, 34.34, 25.78, 15.72, 7.44,
2.21, 147.54, 149.7, 151.04, 130.22, 116.9, 96.59, 86.18, 74.54,
78.3, 77.84, 72.58, 67.87, 59.38, 48.93, 37.82, 26.98, 17.35,
8.21, 3.07, 143.91, 153.27, 156.06, 148.71, 126.64, 122.3, 99.66,
87.97, 74.66, 77.14, 75.86, 68.84, 63.88, 55.07, 41.52, 29.76,
18.92, 9.38, 3.48, 122.45, 139.9, 149.56, 149.18, 132.3, 120.15,
118.2, 96.02, 84.31, 71.51, 73.42, 71.5, 65.21, 58.98, 47.71,
33.94, 21.19, 10.76, 4.92, 121.85, 124, 141.32, 147.86, 137.74,
134.58, 124.39, 119.97, 96.86, 83.01, 70.57, 71.3, 68.44, 60.31,
52.74, 39.68, 24.99, 12.88, 6.03, 140.12, 126.6, 127.66, 143.97,
144.95, 144.49, 144.25, 129.12, 122.83, 97.12, 84.34, 71.68,
72.97, 69.08, 57.87, 46.36, 30.1, 15.36, 6.96, 142.45, 144.23,
130.88, 132.73, 140.22, 145.94, 154.49, 150.34, 133.51, 124.5,
96.24, 82.75, 69.23, 68.72, 62.86, 49.5, 35.57, 18.9, 8.81, 137,
142.8, 147, 136.1, 127.6, 132.6, 152.1, 158.1, 151.9, 132.8,
123.1, 95.3, 81.7, 67.1, 63.3, 54.7, 39.1, 23.5, 11.4, 139.79,
142.65, 150.92, 154.16, 145.33, 133.6, 149.7, 163.43, 166.54,
156.04, 133.72, 122.93, 94.76, 79.91, 62.79, 56.03, 44, 25.92,
14.04, 152.82, 141.16, 143.96, 155.65, 157.45, 149.06, 143.24,
154.06, 164.86, 165.35, 153.29, 131.37, 120.83, 92.33, 75.54,
56.59, 46.44, 30.62, 16.78, 149.99, 153.99, 142.4, 148.3, 159.91,
158.62, 156.34, 146.49, 155.28, 163.73, 162.46, 150.52, 129.04,
117.39, 87.48, 68.56, 47.39, 32.9, 20.57, 148.99, 151.11, 155.14,
146.68, 152.43, 160.8, 165.43, 159.36, 147.63, 154.2, 160.92,
159.63, 147.91, 125.61, 111.45, 79.9, 58.28, 34.28, 23.49, 149.39,
150.12, 152.27, 159.43, 150.83, 153.34, 167.63, 168.47, 160.51,
146.68, 151.57, 158.28, 157.03, 144.17, 119.71, 102.35, 68.66,
43.16, 25.75, 149.53, 150.53, 151.29, 156.57, 163.58, 151.77,
160.21, 170.7, 169.65, 159.57, 144.23, 149.16, 155.92, 153.27,
137.8, 110.67, 88.79, 51.69, 32.32, 149.15, 150.69, 151.71, 155.61,
160.76, 164.53, 158.66, 163.33, 171.91, 168.73, 157.1, 142.04,
147.14, 152.53, 146.87, 128.1, 97.09, 67.84, 40.45, 151, 150.31,
151.88, 156.04, 159.81, 161.72, 171.42, 161.81, 164.6, 171.05,
166.29, 154.87, 140.33, 144.21, 146.65, 137.16, 113.42, 75.6,
53.59, 153.36, 152.17, 151.51, 156.22, 160.27, 160.8, 168.64,
174.59, 163.13, 163.83, 168.67, 164.08, 153.1, 137.85, 139.04,
137.69, 122.31, 89.72, 64.46, 153.69, 154.54, 153.37, 155.87,
160.47, 161.27, 167.73, 171.84, 175.92, 162.41, 161.57, 166.53,
162.34, 150.53, 133.34, 131.08, 123.87, 97.86, 78.52, 153.49,
154.88, 155.74, 157.74, 160.14, 161.48, 168.22, 170.96, 173.2,

```
    175.2, 160.23, 159.6, 164.88, 159.78, 145.89, 126.33, 118.64,
    100.7, 90.41, 153.79, 154.69, 156.08, 160.12, 162.03, 161.16,
    168.45, 171.47, 172.36, 172.54, 173.02, 158.37, 158.19, 162.46,
    155.18, 138.71, 115.3, 97.43, 98.57),
    dim = c(19, 23,  2),
    dimnames = list(
      group = c( "0-4",   "5-9",  "10-14", "15-19", "20-24", "25-29",
                 "30-34", "35-39", "40-44", "45-49", "50-54", "55-59",
                 "60-64", "65-69", "70-74", "75-79", "80-84", "85-89",
                 "90+"),
      year = c("1951", "1956", "1961", "1966",
               "1971", "1976", "1981", "1986",
               "1991", "1996", "2001", "2006",
               "2011", "2016", "2021", "2026",
               "2031", "2036", "2041", "2046",
               "2051", "2056", "2061"),
      sex = c("Male", "Female")))
## Turn array into a more useful form
nzpoplist = list()
for(i in 1:23)
  nzpoplist[[i]] = nzpop[,i,]
names(nzpoplist) = dimnames(nzpop)$year
## Or data.frame
nzpopdf = with(dimnames(nzpop), {
  nzpgroup = factor(rep(group, times = 23 * 2), levels = group)
  nzpyear = rep(rep(as.numeric(year), each = 19), times = 2)
  nzpsex = rep(sex, each = 19 * 23)
  data.frame(freq = as.vector(nzpop),
             group = nzpgroup, year = nzpyear, sex = nzpsex)
})
```