Chapter 1

Graphing Commands

R, and thus Zelig, can produce exceptionally beautiful plots. Many built-in plotting functions exist, including scatter plots, line charts, histograms, bar charts, pie charts, ternary diagrams, contour plots, and a variety of three-dimensional graphs. If you desire, you can exercise a high degree of control to generate just the right graphic. Zelig includes several default plots for one-observation simulations for each model. To view these plots on-screen, simply type `plot(s.out)`, where `s.out` is the output from `sim()`. Depending on the model chosen, `plot()` will return different plots.

If you wish to create your own plots, this section reviews the most basic procedures for creating and saving two-dimensional plots. R plots material in two steps:

1. You must call an output device (discussed in Section **??**), select a type of plot, draw a plotting region, draw axes, and plot the given data. At this stage, you may also define axes labels, the plot title, and colors for the plotted data. Step one is described in Section **??** below.

2. Optionally, you may add points, lines, text, or a legend to the existing plot. These commands are described in Section **??**.

1.1 Drawing Plots

The most generic plotting command is `plot()`, which automatically recognizes the type of R object(s) you are trying to plot and selects the best type of plot. The most common graphs returned by `plot()` are as follows:

1. If `X` is a variable of length n, `plot(X)` returns a scatter plot of (x_i, i) for $i = 1, \ldots n$. If X is unsorted, this procedure produces a messy graph. Use `plot(sort(X))` to arrange the plotted values of (x_i, i) from smallest to largest.

2. With two numeric vectors `X` and `Y`, both of length n, `plot(X, Y)` plots a scatter plot of each point (x_i, y_i) for $i = 1, \ldots n$. Alternatively, if Z is an object with two vectors, `plot(Z)` also creates a scatter plot.

Optional arguments specific to `plot` include:

- `main` creates a title for the graph, and `xlab` and `ylab` label the x and y axes, respectively. For example,

```
plot(x, y, main = "My Lovely Plot", xlab = "Explanatory Variable",
     ylab = "Dependent Variable")
```

- `type` controls the type of plot you request. The default is `plot(x, y, type = "p")`, but you may choose among the following types:

`"p"`	points
`"l"`	lines
`"b"`	both points and lines
`"c"`	lines drawn up to but not including the points
`"h"`	histogram
`"s"`	a step function
`"n"`	a blank plotting region (with the axes specified)

- If you choose `type = "p"`, R plots open circles by default. You can change the type of point by specifying the `pch` argument. For example, `plot(x, y, type = "p", pch = 19)` creates a scatter-plot of filled circles. Other options for `pch` include:

19	solid circle (a disk)
20	smaller solid circle
21	circle
22	square
23	diamond
24	triangle pointed up
25	triangle pointed down

  In addition, you can specify your own symbols by using, for example, `pch = "*"` or `pch = "."`.

- If you choose `type = "l"`, R plots solid lines by default. Use the optional `lty` argument to set the line type. For example, `plot(x, y, type = "l", lty = "dashed")` plots a dashed line. Other options are dotted, dotdash, longdash, and twodash.

- `col` sets the color of the points, lines, or bars. For example, `plot(x, y, type = "b", pch = 20, lty = "dotted", col = "violet")` plots small circles connected by a dotted line, both of which are violet. (The axes and labels remain black.) Use `colors()` to see the full list of available colors.

- `xlim` and `ylim` set the limits to the x-axis and y-axis. For example, `plot(x, y, xlim = c(0, 25), ylim = c(-15, 5))` sets range of the x-axis to $[0, 25]$ and the range of the y-axis to $[-15, 5]$.

For additional plotting options, refer to `help(par)`.

1.2 Adding Points, Lines, and Legends to Existing Plots

Once you have created a plot, you can *add* points, lines, text, or a legend. To place each of these elements, R uses coordinates defined in terms of the x-axes and y-axes of the plot area, not coordinates defined in terms of the the plotting window or device. For example, if your plot has an x-axis with values between $[0, 100]$, and a y-axis with values between $[50, 75]$, you may add a point at $(55, 55)$.

- **points()** plots one or more sets of points. Use `pch` with `points` to add points to an existing plot. For example, `points(P, Q, pch = ".", col = "forest green")` plots each (p_i, q_i) as tiny green dots.

- **lines()** joins the specified points with line segments. The arguments `col` and `lty` may also be used. For example, `lines(X, Y, col = "blue", lty = "dotted")` draws a blue dotted line from each set of points (x_i, y_i) to the next. Alternatively, `lines` also takes command output which specifies (x, y) coordinates. For example, `density(Z)` creates a vector of x and a vector of y, and `plot(density(Z))` draws the kernel density function.

- **text()** adds a character string at the specified set of (x, y) coordinates. For example, `text(5, 5, labels = "Key Point")` adds the label "Key Point" at the plot location $(5, 5)$. You may also choose the font using the `font` option, the size of the font relative to the axis labels using the `cex` option, and choose a color using the `col` option. The full list of options may be accessed using `help(text)`.

- **legend()** places a legend at a specified set of (x, y) coordinates. Type `demo(vertci)` to see an example for `legend()`.

1.3 Saving Graphs to Files

By default, R displays graphs in a window on your screen. To save R plots to file (to include them in a paper, for example), preface your plotting commands with:

```
> ps.options(family = c("Times"), pointsize = 12)
> postscript(file = "mygraph.eps", horizontal = FALSE, paper = "special",
             width = 6.25, height = 4)
```

where the `ps.options()` command sets the font type and size in the output file, and the `postscript` command allows you to specify the name of the file as well as several additional options. Using `paper = special` allows you to specify the width and height of the encapsulated postscript region in inches (6.25 inches long and 4 inches high, in this case), and the statement `horizontal = FALSE` suppresses R's default landscape orientation. Alternatively, you may use `pdf()` instead of `postscript()`. If you wish to select postscript options for .pdf output, you may do so using options in `pdf()`. For example:

```
> pdf(file = "mygraph.pdf", width = 6.25, height = 4, family = "Times",
+     pointsize = 12)
```

At the end of every plot, you should close your output device. The command `dev.off()` stops writing and saves the `.eps` or `.pdf` file to your working directory. If you forget to close the file, you will write all subsequent plots to the same file, overwriting previous plots. You may also use `dev.off()` to close on-screen plot windows.

To write multiple plots to the same file, you can use the following options:

- For plots on separate pages in the same `.pdf` document, use

  ```
  > pdf(file = "mygraph.pdf", width = 6.25, height = 4, family = "Times",
  +     pointsize = 12, onefile = TRUE)
  ```

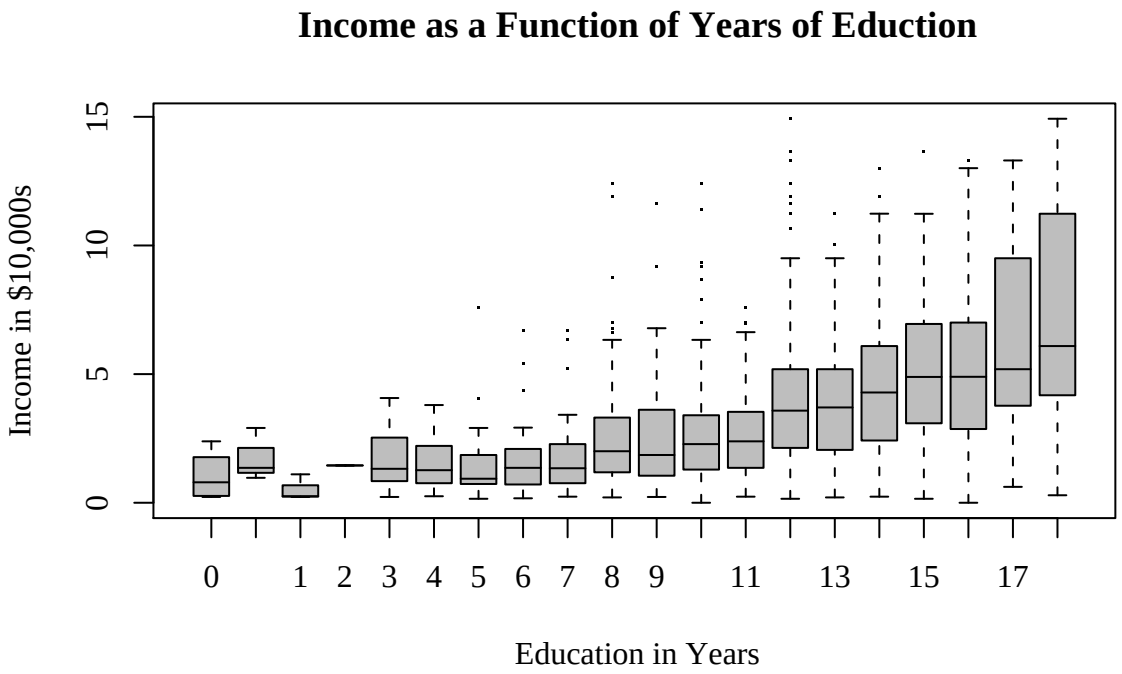
- For multiple plots on one page, initialize either a `.pdf` or `.eps` file, then (before any plotting commands) type:

  ```
  par(mfrow = c(2, 4))
  ```

  This creates a grid that has two rows and four columns. Your plot statements will populate the grid going across the first row, then the second row, from left to right.

1.4 Examples

1.4.1 Descriptive Plots: Box-plots

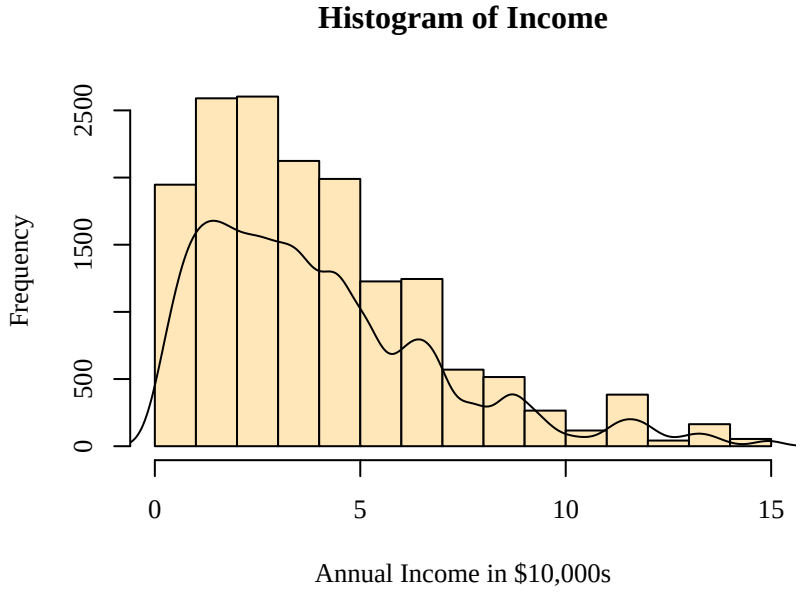


Using the sample `turnout` data set included with Zelig, the following commands will produce the graph above.

```
> library(Zelig)                                # Loads the Zelig package.
> data (turnout)                                # Loads the sample data.
> boxplot(income ~ educate,                     # Creates a boxplot with income
+  data = turnout, col = "grey", pch = ".",     #  as a function of education.
+  main = "Income as a Function of Years of Education",
+  xlab = "Education in Years", ylab = "Income in \$10,000s")
```

1.4.2 Density Plots: A Histogram

Histograms are easy ways to evaluate the density of a quantity of interest.



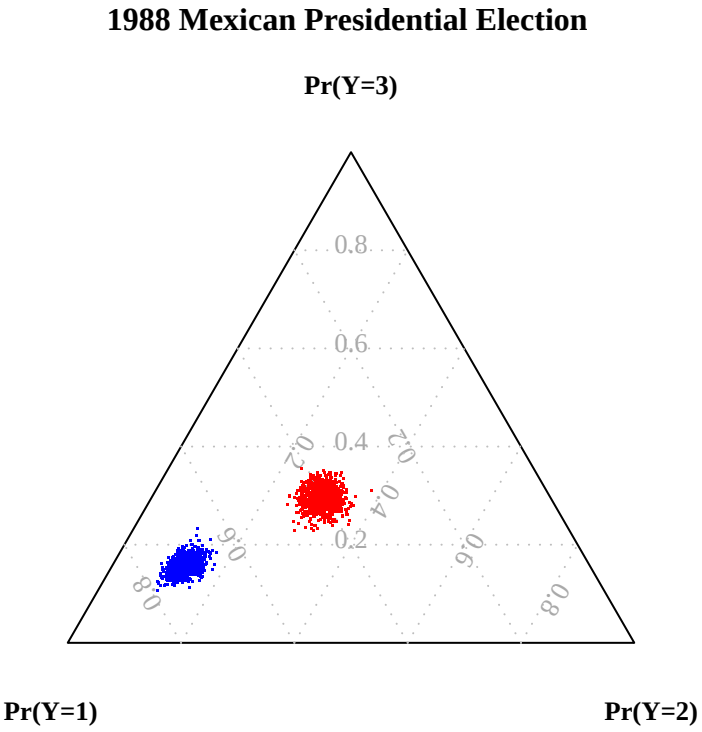
Here's the code to create this graph:

```
> library(Zelig)                              # Loads the Zelig package.
> data(turnout)                               # Loads the sample data set.
> truehist(turnout$income,  col = "wheat1",   # Calls the main plot, with
+      xlab = "Annual Income in $10,000s",    #  options.
+      main = "Histogram of Income")
> lines(density(turnout$income))              # Adds the kernel density line.
```

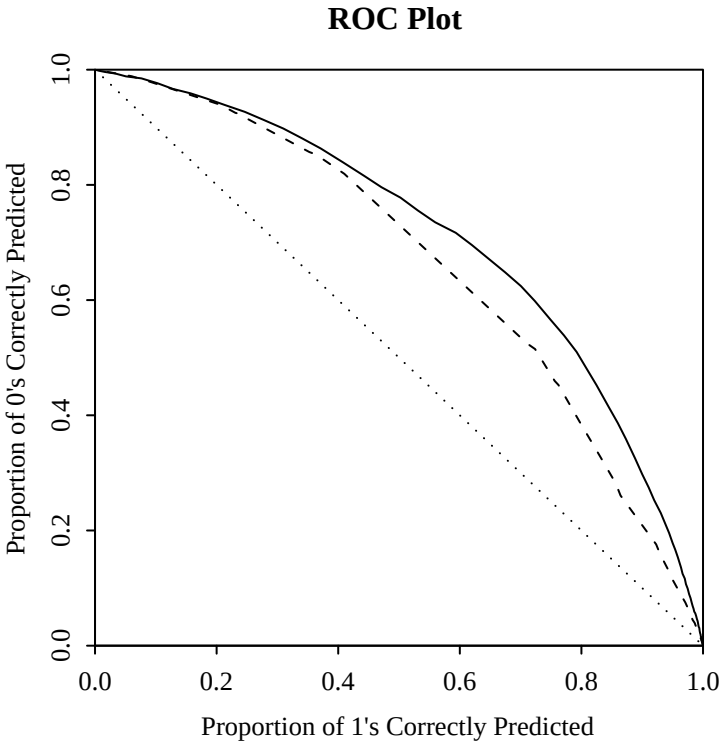
1.4.3 Advanced Examples

The examples above are simple examples which only skim the surface of R's plotting potential. We include more advanced, model-specific plots in the Zelig demo scripts, and have created functions that automate some of these plots, including:

1. **Ternary Diagrams** describe the predicted probability of a categorical dependent variable that has three observed outcomes. You may choose to use this plot with the multinomial logit, the ordinal logit, or the ordinal probit models (Katz and King, 1999). See Section **??** for the sample code, type `demo(mlogit)` at the R prompt to run the example, and refer to Section **??** to add points to a ternary diagram.



2. **ROC Plots** summarize how well models for binary dependent variables (logit, probit, and relogit) fit the data. The ROC plot evaluates the fraction of 0's and 1's correctly predicted for every possible threshold value at which the continuous Prob($Y = 1$) may be realized as a dichotomous prediction. The closer the ROC curve is to the upper right corner of the plot, the better the fit of the model specification (King and Zeng, 2002*b*). See Section **??** for the sample code, and type `demo(roc)` at the R prompt to run the example.



3. **Vertical Confidence Intervals** may be used for almost any model, and describe simulated confidence intervals for any quantity of interest while allowing one of the explanatory variables to vary over a given range of values (King, Tomz and Wittenberg, 2000). Type `demo(vertci)` at the R prompt to run the example, and `help.zelig(plot.ci)` for the manual page.