Chapter 2

Exploring Data

2.1 What Is Statistics?

You may be wondering "What is statistics?", "Who uses it?", and "Why do I need to study this material?" Statistics is the process of finding out more about a topic by collecting information and then trying to make sense out of that information. In essence, statistics is concerned with methods for collecting, organizing, summarizing, presenting, and analyzing data. Data laden information is present in virtually every sector of society, and the need to make sense out of our surroundings is a basic human need. More to the point of why you, the reader, might need to study this material can be answered in one of two ways. First, you are required to study this material as part of your major because there are certain topics that are deemed important by your teachers. Second, you desire to have some modicum of control in decision making and want to learn more about how probability and statistics help people, corporations, and governmental agencies make decisions/policies. Even if your reason for reading this material is because it is required, it is a fervent hope that your ability to make sound decisions is strengthened through the material in this book.

2.2 Data

Data, according to *The American Heritage Dictionary*, are "Information, especially information organized for analysis or used as the basis for a decision." A characteristic that is being studied in a statistical problem is called a **variable**. A variable will be either **qualitative** or **quantitative**. When a variable is qualitative, it is essentially defining groups or categories. When the categories have no ordering the variable is called **nominal**. For example, the variable gender can take on the values male and female or the variable "music preference" could have values such as "classical," "jazz," "rock," or "other." When the categories have a distinct ordering, the variable is called **ordinal**. Such a variable might be educational level with values elementary school, high school, college graduate, graduate or professional school. Values on a scale can be either interval or ratio. Interval data have interpretable distances, while ratio data have a true zero. A variable that is quantitative (numeric) may be either **discrete** or **continuous**. A discrete variable is a numerical variable that can assume a finite number or at most a countably infinite number of values. Such variables include the number of people arriving at a bank on Thursday, students in a class, or dogs in the pound. A continuous variable is a numerical variable that can assume an infinite number of values associated with the numbers on an interval of the real number line, for example, the height of a tree, the life of a light bulb, the weight of an apple. An important distinction between discrete and continuous variables is that discrete variables can take on the same value repeatedly while continuous variables have few or no repeated values. It is important to be able to distinguish between different types of variables since methods for viewing and summarizing data are dependent on variable type. More to the point, it will be imperative to distinguish between qualitative (categorical) variables and quantitative (numerical) variables.

When a data set consists of a single variable, it is called a **univariate** data set. When there are two variables in the data set, the data is set is called a **bivariate** data set; and when there are two or more variables, the data set is called a **multivariate** data set. In the remainder of this section, the discussion will cover univariate variables. Recall that a qualitative variable defines categories or groups. The membership in these categories is summarized with tables and graphically illustrated with bar graphs.

2.3 Displaying Qualitative Data

2.3.1 Tables

A table that lists the different groups of categorical data and the corresponding frequencies with which they occur is called a **frequency table**. Qualitative information is typically presented in the form of a frequency table. The S function `table()` can be used to create various types of tables.

Example 2.1 Suppose the letter grades of an English essay in a small class are A, D, C, D, C, C, C, C, F, and B. Create both a frequency table showing the numbers and a relative frequency table showing the proportions of the various grades.

Solution: First, the character data are read into a vector named `Grades`. Then, the S function `table()` is applied to `Grades`:

```
> Grades <- c("A","D","C","D","C","C","C","C","F","B")
> Grades
 [1] "A" "D" "C" "D" "C" "C" "C" "C" "F" "B"
> table(Grades)
Grades
A B C D F
1 1 5 2 1
> table(Grades)/10          # Relative frequency table
Grades
  A   B   C   D   F
0.1 0.1 0.5 0.2 0.1
```

Clearly, there is no need for a computer with such a small data set; however, tables for much larger data sets can be created with no more work than that required for this small data set. ■

Example 2.2 The `quine` data frame in the `MASS` package has information on children from Walgett, New South Wales, Australia, that were classified by `Culture`, `Age`, `Sex`, and `Learner` status including the number of `Days` absent from school in a particular school year. Use the function `table()` to create a frequency table for the variable `Age`.

Solution: To gain access to information stored in `MASS`, first load the package and attach the data frame `quine`:

```
> library(MASS)
> attach(quine)
> table(Age)
Age
F0 F1 F2 F3
27 46 40 33
```

■

2.3.2 Barplots

One of the better graphical methods to summarize categorical data is with a **barplot**. Barplots are also known as bar charts or bar graphs. The S function `barplot()` is used to create barplots using a summarized version of the data, often the result of the `table()` function. This summarized form of the data can be either frequencies or percentages. Regardless of whether one uses frequencies or percentages, the resulting shape looks identical, but the scales on the y-axes are different.

Example 2.3 Construct barplots for the variables `Grades` used in Example 2.1 and `Age` in the `quine` data set from the `MASS` package in Example 2.2 on the facing page using both frequencies and proportions.

Solution: Before creating any barplots, the device region is split into four smaller regions with the command `par(mfrow=c(2,2))`:

```
> par(mfrow=c(2,2))
> barplot(table(Grades), col=3, xlab="Grades", ylab="Frequency")
> barplot(table(Grades)/length(Grades), col=3, xlab="Grades", ylab=
+ "Proportion")
> barplot(table(Age), col=7, xlab="Age", ylab="Frequency")
> barplot(table(Age)/length(Age), col=7, xlab="Age", ylab="Proportion")
```

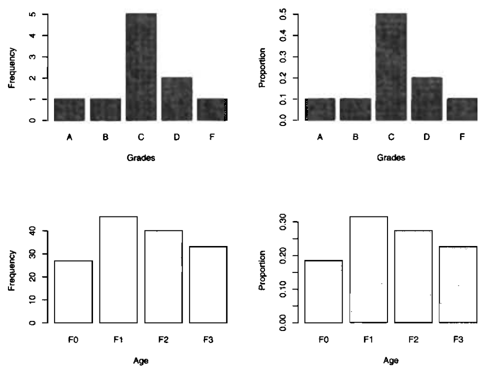

FIGURE 2.1: Graphical representation of the data in `Grades` and `Age` with the function `barplot()` ■

2.3.3 Dot Charts

An equally effective way to display qualitative data is by using a **dot chart**. Dot charts are also called Cleveland dotplots. A dot chart shows the values of the variables of interest (levels of the qualitative variable) as dots in a horizontal display over the range of the data. The S command to create a dot chart is `dotchart(data)`, where `data` is a vector containing frequencies for all the different levels of a variable. When working with un-summarized data, one way to prepare the data for a `dotchart()` is first to summarize the data with the command `table()`. The optional arguments for `dotchart()` in R and S-PLUS are different, and the user should consult the respective documentation for further assistance.

Example 2.4 Construct dot charts for the variables `Grades` from Example 2.1 and `Age` used in the `quine` data set from the `MASS` package in Example 2.2 on page 30.

Solution: Before creating any dot charts, the device region is split into two smaller regions with the command `par(mfrow=c(2,1))`:

```
> par(mfrow=c(1,2))
> dotchart(table(Grades))
> dotchart(table(Age))
```

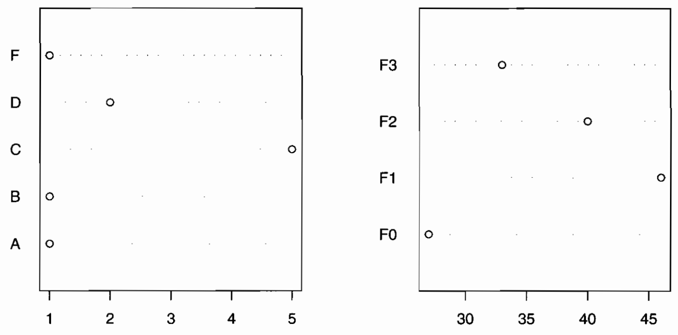

FIGURE 2.2: Graphical representation of the data in `Grades` and `Age` with the function `dotchart()` ■

2.3.4 Pie Charts

Pie charts represent the relative frequencies or percentages of the levels of a categorical variable with wedges of a pie (circle). While the media often use **pie charts** to display qualitative data, the pie chart has fallen out of favor with most statisticians. Pie charts are most useful when the emphasis is on each category in relation to the total. When such an emphasis is not the primary point of the graphic, a bar chart or a dot chart should be used.

Example 2.5 Construct pie charts for the variables `Grades` in Example 2.1 and `Age` from the `quine` data set in the `MASS` package used in Example 2.2 on page 30.

Solution: Before creating any pie charts, the device region is split into two regions with the command `par(mfrow=c(2,1))`:

```
> par(mfrow=c(1,2))
> pie(table(Grades))
> title("Grades")
> pie(table(Age))
> title("Age")
```

The graph depicted in Figure 2.3 was produced in R with the additional arguments `radius=2.5` and `col=gray(c(.1,.4,.7,.8,.95))`.

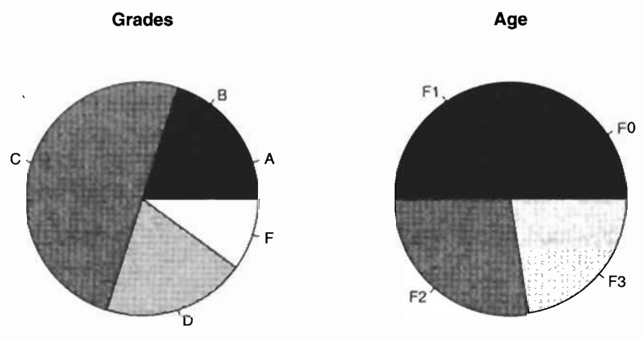

FIGURE 2.3: Graphical representation of the data in `Grades` and `Age` with the function `pie()` ■

2.4 Displaying Quantitative Data

When presented with quantitative data, knowing three facts about the data, namely, its shape, center, and spread, will be a great start in making some sense of the numbers. Some of the more common distribution shapes are shown in Figure 2.4 on the following page. Of the nine different shapes in Figure 2.4, all are symmetric with the exception of the second and the eighth graphs, which are characterized as skewed to the right and skewed to the left, respectively. Of the nine different shapes in Figure 2.4, all are unimodal with the exception of the first, the fourth, and the ninth graphs, which are characterized as bimodal, uniform, and multi-modal, respectively. One final highlight: When presented with a symmetric unimodal data set, it will be important to classify the distribution as either short-tailed, long-tailed, or normal. The fourth and the sixth graphs, in addition to being symmetric, are also short-tailed. What follows are graphical tools that can help in assessing the shape, center, and spread of a data set. As a general rule, the shape of the data dictates the most appropriate measures of center and spread for that data set.

2.4.1 Stem-and-Leaf Plots

One way to get a quick impression of the data is to use a **stem-and-leaf plot**. When a stem-and-leaf plot is constructed, each observation is split into a stem and a leaf. Regardless of where the observation is split, the leaf in a stem-and-leaf plot is represented with a single

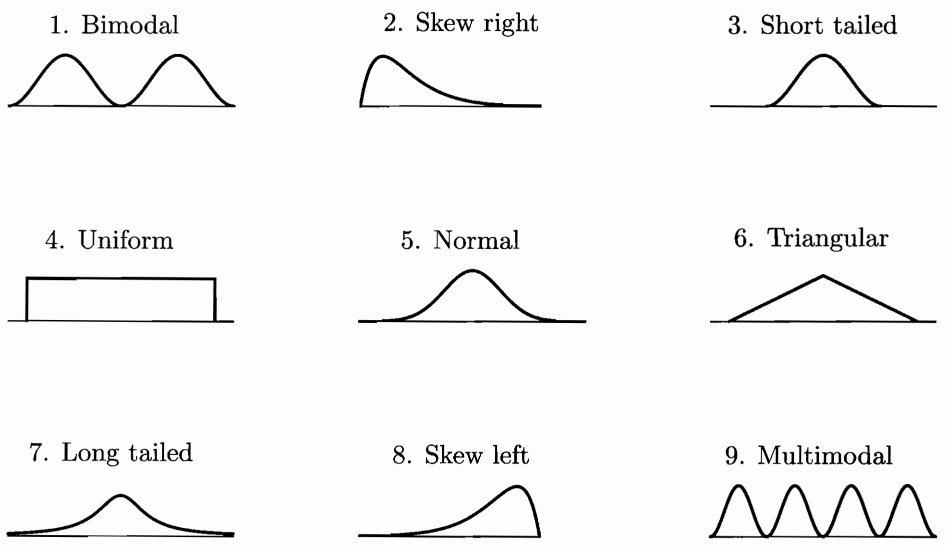

FIGURE 2.4: Nine different graphs labeled according to their shape

digit. Although it is possible to use a stem-and-leaf plot with a moderately sized data set (more than 100 values), the plot becomes increasingly hard to read as the number of values plotted increases. Consequently, it is recommended that stem-and-leaf plots be used graphically to illustrate smallish data sets (less than 100 values). The S command to create a stem-and-leaf plot is `stem(x)`, where `x` is a numeric vector.

Example 2.6 Use the data frame `Baberuth` to construct a stem-and-leaf plot for the number of home runs (`HR`) Babe Ruth hit while he played for the New York Yankees.

Solution: A quick glance at the data frame `Baberuth` shows that Babe Ruth played for the New York Yankees for his seventh through twenty-first seasons. The information in `HR` is for Babe Ruth's entire (22 seasons) professional career. To extract the home runs he hit while he was a New York Yankee, use `HR[Team=="NY-A"]` or `HR[7:21]` (seventh through twenty-first season home runs):

```
> attach(Baberuth)             # Assumes package PASWR is loaded
> NYYHR <- HR[Team=="NY-A"]
> NYYHR
 [1] 54 59 35 41 46 25 47 60 54 46 49 46 41 34 22
> stem(NYYHR)

  The decimal point is 1 digit(s) to the right of the |

  2 | 25
  3 | 45
  4 | 1166679
  5 | 449
  6 | 0

> detach(Baberuth)
```

In this example, see how the stems 2–6 represent the values twenty through sixty and the leaves represent the second digit of the numbers in HR. Reading the first row of the stem-and-leaf plot notice the values 22 and 25. The stem-and-leaf plot reveals a fairly symmetric distribution. ■

2.4.2 Strip Charts (R Only)

An alternative to the stem-and-leaf plot is a **strip chart** (also referred to as a dotplot by many authors). A strip chart plots values along a line. The R function `stripchart()` will stack the tied observations in a column at each value observed along a line that covers the range of the data when given the argument `method="stack"`. The function requires the data to be a vector, a list of vectors, or a formula of the form x~g, where values are in a vector x and groups are in a vector g. Strip charts are often useful for comparing the distribution of a quantitative variable at different qualitative levels (groups).

Example 2.7 Use the data frame Baberuth to

(a) Construct a strip chart of the number of home runs Babe Ruth hit while playing for the New York Yankees.

(b) Create a strip chart of the number of home runs Babe Ruth hit per season according to the team for which he was playing. Based on the strip chart, when Babe Ruth played, for which team did he generally hit more home runs per season?

Solution: (a) Figure 2.5 on the next page is a strip chart of the number of home runs Babe Ruth hit while playing for the New York Yankees. The code to construct this graph is

```
> attach(Baberuth)
> Baberuth[1:5,]        # equivalently heads(Baberuth, n=5)
  Year  Team  G  AB  R  H X2B X3B HR RBI SB BB    BA   SLG
1 1914 Bos-A  5  10  1  2   1   0  0   0  0  0 0.200 0.300
2 1915 Bos-A 42  92 16 29  10   1  4  21  0  9 0.315 0.576
3 1916 Bos-A 67 136 18 37   5   3  3  16  0 10 0.272 0.419
4 1917 Bos-A 52 123 14 40   6   3  2  12  0 12 0.325 0.472
5 1918 Bos-A 95 317 50 95  26  11 11  66  6 58 0.300 0.555
> NYYHR <- HR[7:21]      # Extracts the 7th through 21st season HR values.
> stripchart(NYYHR, xlab="Home runs per season", pch=1, method="stack",
+ main="Dotplot of home runs while a New York Yankee")
```

(b) Figure 2.6 on the following page is a strip chart of the number of home runs Babe Ruth hit per season according to the team for which he was playing. The code to construct this graph is

```
> par(mfrow=c(1,2), pty="s")
> stripchart(HR~Team, pch=1, method="stack",
+ main="Dotplot of home runs \n by team",
+ xlab="Home runs per season")
> par(las=1)   # Makes labels horizontal
> stripchart(HR~Team, pch=19, col=c("red","green","blue"),
+ method="stack", main="Color dotplot of home runs \n by team",
+ xlab="Home runs per season")
> par(mfrow=c(1,1), las=0, pty="m")
> detach(Baberuth)
```

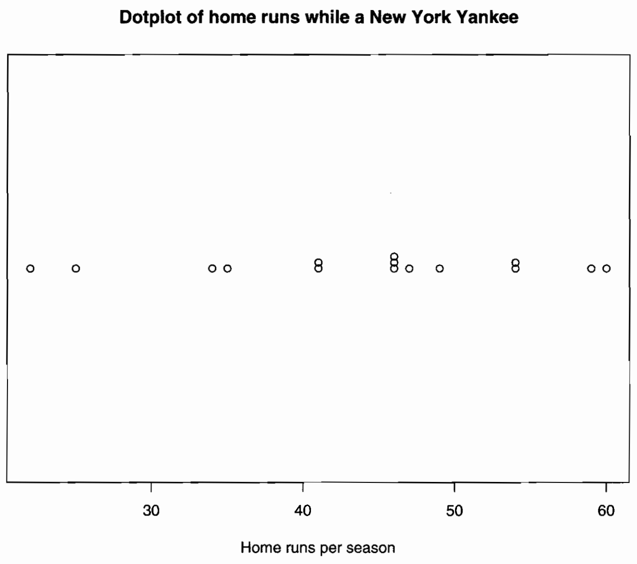

FIGURE 2.5: Strip chart of the number of home runs Babe Ruth hit while playing for the New York Yankees

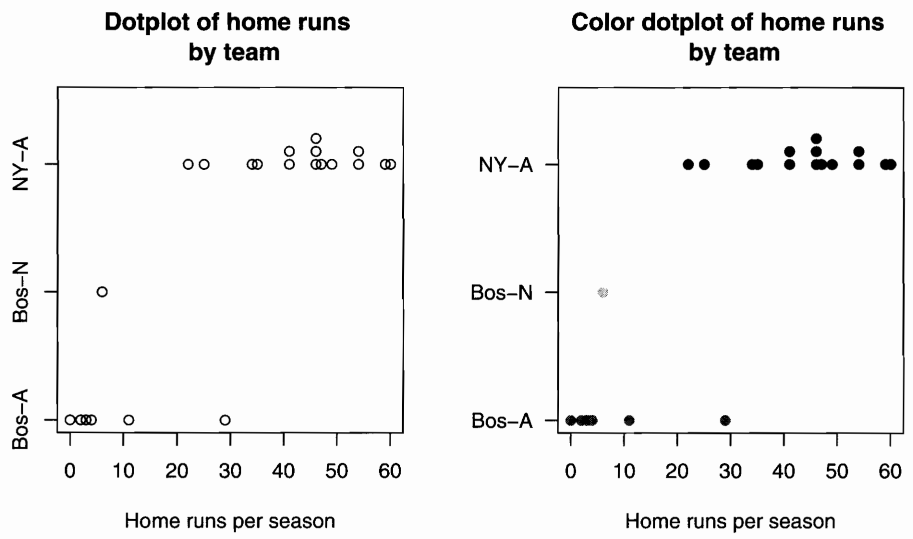

FIGURE 2.6: Strip chart of the number of home runs Babe Ruth hit per season according to the team for which he was playing

2.4.3 Histograms

The **histogram** is a graphical means of illustrating quantitative (numerical) data. Although the barplot and the histogram look similar, the barplot is used for qualitative data while the histogram is used for numerical data. Yet, the bins that either the user specifies or those that S uses by default are in essence categories. Histograms created in S with the function `hist(x)`, where `x` is a numeric vector, are by default frequency histograms. To create density histograms, use the optional argument `prob=TRUE`. A density histogram has a total area of one.

Example 2.8 Construct a histogram that resembles the stem-and leaf plot from Example 2.6 using the `Baberuth` data.

Solution: The first histogram uses the default arguments for `hist()`. Since the bins S uses are of the form `(]`, the default histogram does not resemble the stem-and-leaf plot. To change the bins to the form `[)` in R, use the argument `right=FALSE`:

```
> attach(Baberuth)
> par(mfrow=c(1,2))
> bin <- seq(20,70,10)    # Creating bins 20-70 by 10
> hist(HR[7:21], breaks=bin, xlab="Home Runs")
> hist(HR[7:21], breaks=bin, right=FALSE, xlab="Home Runs") # R
> detach(Baberuth)
```

The graph depicted in Figure 2.7 was produced in R with commands similar to those given. One way to produce the second graph in S-PLUS is to use a slight fudge factor when creating the bins, such as `bin <- seq(20,70,10)-0.00001`.

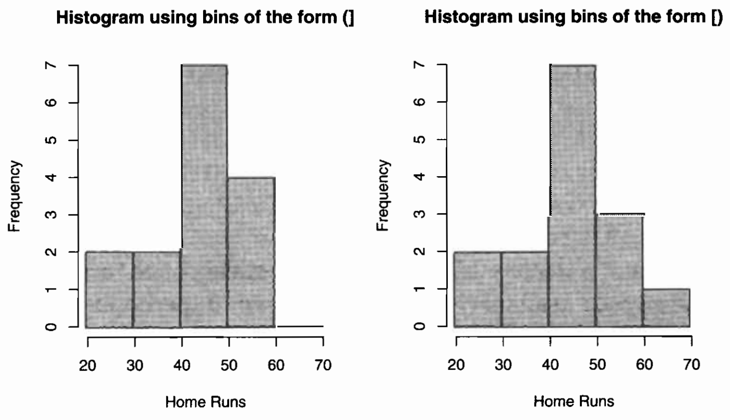

FIGURE 2.7: Histograms created using different bin definitions for the number of home runs hit by Babe Ruth while playing for the New York Yankees ■

One of the problems with using histograms to describe the shape of the data is the arbitrary nature of the bin width. In Example 2.8, it was seen how simply including or excluding an end point changed the histogram. Consider the differences among the shapes of the histograms in Figure 2.8 on the next page produced by simply altering the bin width. The data set used to produce Figure 2.8 on the following page is `geyser`, available in the `MASS` package. A much better choice to get an idea of what the shape of a distribution looks like is to use a **density estimate**. The S function `density(x)`, where `x` is a numeric vector, can be used to create a density estimate. Basically, a density estimate uses shapes with $\frac{1}{n}$ area added up at each point in the data set to create a graph with area 1. The resulting shape is a density estimate. The result of the density estimate can be viewed with either the `plot()` or `lines()` function. Recall that `plot()` is a high-level function while `lines()` is a low-level function. That is, `plot()` will create a graph while `lines()` will add to an existing graph.

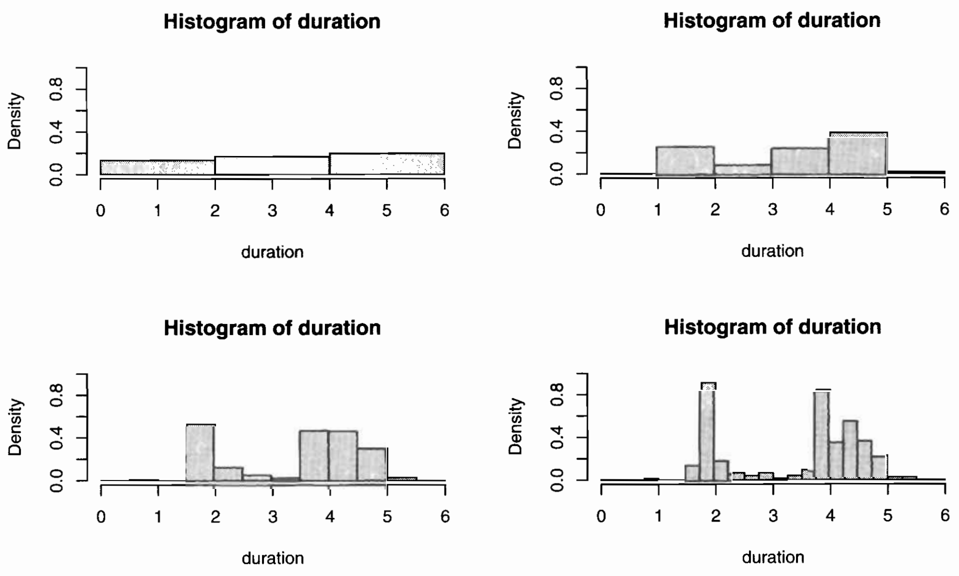

FIGURE 2.8: Histograms created using different bin definitions for the eruption duration of Old Faithful

Example 2.9 Construct a density histogram of the waiting time until the next eruption using the data frame `geyser` available in the `MASS` package. Superimpose a density estimate over the density histogram. In the same graph, show the estimated density without showing the histogram.

Solution: Note that to superimpose a density over a histogram, the histogram must be a density histogram. Recall that density histograms are produced with the optional argument `prob=TRUE`:

```
> library(MASS)
> par(mfrow=c(1,2))       # Make device region 1 by 2
> attach(geyser)
> hist(waiting, prob=TRUE)
> lines(density(waiting)) # Add density to Histogram
> plot(density(waiting))  # Create density by itself
> detach(geyser)
```

Based on the density estimates, it appears there are two modes for waiting time until the next eruption. It seems one will usually have to wait close to either 50 or 80 minutes until the next eruption. ■

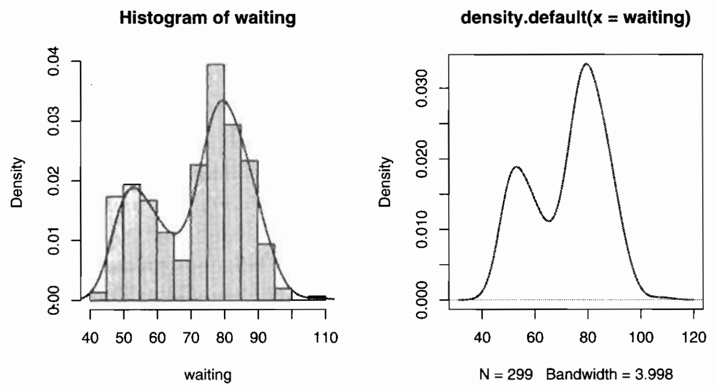

FIGURE 2.9: Histogram of waiting time between Old Faithful eruptions with superimposed density estimate as well as a density plot

2.5 Summary Measures of Location

One of the main objectives of statistics is to make inference to a population based on information obtained from a sample. Since it can be overwhelming to work with the entire population and/or sample, summary measures are introduced to help characterize the data at hand. These summary measures may apply to either the population or to the sample. Numerical summaries of the population are called **parameters** while numerical summaries of the sample are called **statistics**. More formal definitions of both parameters and statistics will be given later. Measures of central location are introduced first. The measures covered are generally familiar to the reader from everyday usage. Specifically, the **mean**, the **trimmed mean**, and the **median** are introduced. Other measures of location addressed include quartiles, hinges, and quantiles.

2.5.1 The Mean

The most common measure of center is the average, which locates the balance point of the distribution or data. The mean is an appropriate measure of center for symmetric distributions; however, it is not appropriate for skewed distributions. In statistics, the average of a sample is called the **sample mean** and is denoted by $\bar{x}$. Given some numeric data $x_1, x_2, \ldots, x_n$, the sample mean is defined as

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \sum_{i=1}^{n} \frac{x_i}{n} \tag{2.1}$$

The S function `mean(x)` will compute the mean of a data vector `x`. Additional arguments to `mean(x)` include `na.rm=TRUE`, for removal of missing values, and `trim=`, to compute a trimmed mean. The trimmed mean is generally used to estimate the center when working with long-tailed distributions. When a $p\%$ trimmed mean is computed, $p\%$ of the sorted data is deleted from each end of the distribution, and a mean is computed from the remaining values. When $p \times n$ is not an integer, the integer portion, ($\lfloor p \times n \rfloor$), should be deleted from each end of the sorted values and the mean computed from the remaining values.

Example 2.10 Compute the mean number of home runs per season Babe Ruth hit while playing for the New York Yankees. Compute a 5%, a 10%, a 15%, and a 50% trimmed mean for the number of home runs per season Babe Ruth hit while playing for the New York Yankees using the information stored in the data frame `Baberuth`.

Solution: In Example 2.6 on page 34, the variable `NYYHR` was created that contained the number of home runs Babe Ruth made while playing for the New York Yankees. If `NYYHR` is no longer available, recreate it with the command `NYYHR <- HR[7:21]` once the data frame `Baberuth` has been attached. Since there are 15 values in `NYYHR`, to compute 5%, 10%, 15%, and 50% trimmed means, $\lfloor 0.05 \times 15 \rfloor = \lfloor 0.75 \rfloor = 0$, $\lfloor 0.10 \times 15 \rfloor = \lfloor 1.5 \rfloor = 1$, $\lfloor 0.15 \times 15 \rfloor = \lfloor 2.25 \rfloor = 2$, and $\lfloor 0.50 \times 15 \rfloor = \lfloor 7.5 \rfloor = 7$ values, respectively, will need to be deleted from the sorted values of `NYYHR` before computing means on the remaining values. A second solution is also presented using the S function `mean()` using the `trim=` argument:

```
> attach(Baberuth)
> NYYHR <- HR[7:21]
> NYYHR
 [1] 54 59 35 41 46 25 47 60 54 46 49 46 41 34 22
> SNYYHR <- sort(NYYHR)
> SNYYHR
 [1] 22 25 34 35 41 41 46 46 46 47 49 54 54 59 60
> p.05 <- floor(.05*15)
> p.10 <- floor(.10*15)
> p.15 <- floor(.15*15)
> p.50 <- floor(.50*15)
> num.to.delete <-c(p.05, p.10, p.15, p.50)
> num.to.delete
[1] 0 1 2 7
> m.05 <- mean(SNYYHR[(1+p.05):(15-p.05)])
> m.10 <- mean(SNYYHR[(1+p.10):(15-p.10)])
> m.15 <- mean(SNYYHR[(1+p.15):(15-p.15)])
> m.50 <- mean(SNYYHR[(1+p.50):(15-p.50)])
> t.m <- c(m.05, m.10, m.15, m.50)
> names(t.m) <- c("5%tmean","10%tmean","15%tmean","50%tmean")
> t.m
 5%tmean 10%tmean 15%tmean 50%tmean
43.93333 44.38462 44.81818 46.00000
> tm.05 <- mean(NYYHR, trim=.05)
> tm.10 <- mean(NYYHR, trim=.10)
> tm.15 <- mean(NYYHR, trim=.15)
> tm.50 <- mean(NYYHR, trim=.50)
> tms <- c(tm.05, tm.10, tm.15, tm.50)
> names(tms) <- c("5%tmean","10%tmean","15%tmean","50%tmean")
> tms
 5%tmean 10%tmean 15%tmean 50%tmean
43.93333 44.38462 44.81818 46.00000
> detach(Baberuth)
```

The trimmed means are all fairly similar, confirming a rather symmetric distribution. Note that the 50% trimmed mean is the value in the middle of the sorted observations. This value is also known as the median. ■

2.5.2 The Median

While the mean is the most commonly encountered measure of center, it is not always the best measure of center. The **sample median** is the middle value of a distribution of numbers, denoted by the letter m. Since the median ignores the information in surrounding values, it is more resistant to extreme fluctuations in the data than is the mean. When working with skewed distributions, the median is the most appropriate measure of center. The sample median, m, of $x_1, x_2, \ldots, x_n$ is the $\left(\frac{n+1}{2}\right)^{\text{st}}$ observation of the sorted values. When n is odd, $\frac{n+1}{2}$ is an integer, and finding the observation is straightforward. When n is even, an average of the two middle observations is taken to find the median. When the values $x_1, x_2, \ldots, x_n$ are sorted, they are called **order statistics** and denoted as $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$. A more concise definition of the sample median is then

$$m = \begin{cases} x_{(k+1)} & n = 2k+1 \text{ (odd)}, \\ \frac{1}{2}(x_{(k)} + x_{(k+1)}) & n = 2k \text{ (even)}. \end{cases} \tag{2.2}$$

To find the sample median with S use the function `median(x)`, where `x` is a numeric vector.

Example 2.11 ▷ ***Means and Medians*** ◁ The numerical grades achieved by three students on four exams during the course of a semester are recorded in Table 2.1. Compute means and medians for the students. Could the three students be characterized?

Table 2.1: Student test scores

	Test1	Test2	Test3	Test4
Student1	73	75	74	74
Student2	95	94	12	95
Student3	66	67	63	100

Solution: First the students exam scores are read into individual vectors denoted `Student1`, `Student2`, and `Student3`. The S function `median()` is used first to find the median test score for each student. It is possible to compute the mean test score for each student in a similar fashion to that used to find the median test score for each student. However, another solution is provided by using the S functions `rbind()`, `cbind()`, and `apply()`:

```
> Student1 <- c(73,75,74,74)
> Student2 <- c(95,94,12,95)
> Student3 <- c(66,67,63,100)
> median(Student1)
[1] 74
> median(Student2)
[1] 94.5
> median(Student3)
[1] 66.5
> SM <- rbind(Student1, Student2, Student3)  # combine rows
> colnames(SM) <- c("Test1","Test2","Test3", "Test4")
> SM
         Test1 Test2 Test3 Test4
Student1    73    75    74    74
Student2    95    94    12    95
Student3    66    67    63   100
> means <- apply(SM,1, mean)      # mean of rows
> medians <- apply(SM,1, median)  # median of rows
> TOT <- cbind(SM, means, medians) # combine columns
> TOT
         Test1 Test2 Test3 Test4 means medians
Student1    73    75    74    74    74    74.0
Student2    95    94    12    95    74    94.5
Student3    66    67    63   100    74    66.5
```

As seen in the S output, the mean test score for the three students is 74. One possible characterization of the three students might be: Student 1: consistent; Student 2: overconfident; Student 3: procrastinator. Would the mean or the median be the better representative in assigning their final grades? There are good reasons one may want to consider using the median instead of the mean. ■

2.5.3 Quantiles

The **p^{th} quantile**, $0 \le p \le 1$, of a distribution is the value x_p such that $\mathbb{P}(X \le x_p) \ge p$ and $\mathbb{P}(X \ge x_p) \ge 1-p$. For discrete data, there are often many values of x_p that satisfy the definition of the p^{th} quantile. In this book, the definition used by S to compute quantiles will be used. S defines the p^{th} quantile of a distribution to be the $\big(p(n-1)+1\big)^{\text{st}}$ order statistic. When $p(n-1)+1$ is not an integer, linear interpolation is used between order statistics to arrive at the p^{th} quantile. Given values $x_1, x_2, \ldots, x_n$, the p^{th} quantile for the k^{th} order statistic, $p(k)$, is

$$p(k) = \frac{(k-1)}{(n-1)}, \quad k \le n. \tag{2.3}$$

By this definition, it is seen that the 50% quantile (50th percentile) is the median since

$$0.50 = \frac{k-1}{n-1} \Rightarrow k = \frac{n+1}{2},$$

which by definition is the location of the order statistic that is the median. Other definitions for quantiles exist and are used in other texts and other statistical software packages. However, the definition used here is consistent with S-PLUS and the default algorithm used in R for computing quantiles. To read about alternative algorithms for computing quantiles with R, type `?quantile` at the R prompt. To compute the quantiles of a data set stored in a vector `x`, use the S function `quantile(x)`. By default, the S function `quantile(x)` returns the 0%, 25%, 50%, 75%, and 100% quantiles of the data vector `x`. The p^{th} quantile is the same thing as the $(p \times 100)^{\text{th}}$ percentile. That is, percentiles and quantiles measure the same thing; however, percentiles use a scale from 0 to 100 instead of the 0 to 1 scale used by quantiles.

Just as the sample median is the value that divides the sample into equal halves, the **sample quartiles** can be thought of as the values that divide the sample into quarters. The first, second, and third sample quartiles are denoted as Q_1, Q_2, and Q_3, respectively, and are (by default) computed with the S function `quantile(x)`. To compute other quantiles, use the argument `probs=` to specify either a single value or to pass a vector of values to the `quantile()` function.

Example 2.12 Compute Q_1, Q_2, and Q_3 for the values $x_{(1)} = 1$, $x_{(2)} = 4$, $x_{(3)} = 7$, $x_{(4)} = 9$, $x_{(5)} = 10$, $x_{(6)} = 14$, $x_{(7)} = 15$, $x_{(8)} = 16$, $x_{(9)} = 20$, and $x_{(10)} = 21$.

Solution: First, the order statistics for the 0.25, 0.50, and 0.75 quantiles are computed using (2.3):

$$.25 = \frac{k-1}{10-1} \qquad .50 = \frac{k-1}{10-1} \qquad .75 = \frac{k-1}{10-1}$$
$$k = 3.25 \qquad k = 5.50 \qquad k = 7.75$$

Linear interpolation is then used on the order statistics to find the requested quantiles/quartiles. Specifically, since Q_1, Q_2, and Q_3 occur at the 3.25, 5.50, and 7.75 order statistics, 0.25 of the distance between the third and fourth order statistics is added to the third order statistic to arrive at Q_1. Likewise, 0.50 of the distance between the fifth and sixth order statistics is added to the fifth order statistic to compute Q_2. Finally, 0.75 of the distance between the seventh and eighth order statistics is added to the seventh order statistic to compute Q_3:

$$\begin{aligned} Q_1 &= x_{(3)} + .25(x_{(4)} - x_{(3)}) \\ &= 7 + .25(9-7) \\ &= 7.50 \end{aligned} \qquad \begin{aligned} Q_2 &= x_{(5)} + .50(x_{(6)} - x_{(5)}) \\ &= 10 + .5(14-10) \\ &= 12.00 \end{aligned}$$

$$\begin{aligned} Q_3 &= x_{(7)} + .75(x_{(8)} - x_{(7)}) \\ &= 15 + .75(16-15) \\ &= 15.75 \end{aligned}$$

Code to compute the requested quartiles according to the quantile definition follows. Subsequently, the S function `quantile()` is used to compute the same quantiles/quartiles.

```
> x <- c(1,4,7,9,10,14,15,16,20,21)
> p <- c(.25,.5,.75)              # desired quantiles
> n <- length(x)                  # number of values, n
> order.stat <- p*(n-1)+1         # computing order statistics
> order.stat                      # order statistics
[1] 3.25 5.50 7.75
> Q1 <- x[3]+.25*(x[4]-x[3])      # linear interpolation
> Q2 <- x[5]+.50*(x[6]-x[5])      # linear interpolation
> Q3 <- x[7]+.75*(x[8]-x[7])      # linear interpolation
> QU <- c(Q1, Q2, Q3)
> names(QU) <- c("Q1","Q2","Q3")
> QU                              # quartiles
   Q1    Q2    Q3
 7.50 12.00 15.75
> quantile(x, probs=c(.25,.5,.75))  # the easy way!
  25%   50%   75%
 7.50 12.00 15.75
```

■

2.5.4 Hinges and Five-Number Summary

An alternative method to calculating quartiles is to compute **hinges**. The idea behind both quartiles and hinges is to split the data into fourths. When a computer is not available, hinges are somewhat easier to compute by hand than are quartiles. The lower and upper hinges are the $x_{(j)}$ and $x_{(n-j+1)}$ order statistics, where

$$j = \frac{\lfloor \frac{n+1}{2} \rfloor + 1}{2}. \tag{2.4}$$

In short, the lower hinge is the median of the lower half of the data and the upper hinge is the median of the upper half of the data. Lower and upper hinges can be different from quartiles. For example, consider Example 2.12 on the previous page where the locations of the first, and third quartiles were found to be at the 3.25th and 7.75th order statistics. However, since

$$\frac{\lfloor \frac{n+1}{2} \rfloor + 1}{2} = \frac{\lfloor \frac{10+1}{2} \rfloor + 1}{2} = 3,$$

the locations for the lower and upper hinges are at the 3rd, $x_{(3)}$, and 8th, $x_{(n-3+1)} = x_{(10-3+1)} = x_{(8)}$, order statistics.

Hinges are typically returned as part of the **five-number summary**. A five-number summary for a data set consists of the smallest value, the lower hinge, the median, the upper hinge, and the largest value, all of which are computed with R's function `fivenum()`.

Example 2.13 Compute the 0.25, 0.50, and 0.75 quantiles as well as a five-number summary for the number of runs batted in (RBIs) by Babe Ruth while he played for the New York Yankees. The variable `RBI` in the data frame `Baberuth` contains the RBIs per season for Babe Ruth over his professional baseball career.

Solution: The quartiles and hinges are first computed by their definitions. Subsequently, the S function `quantile()` and the R function `fivenum()` are used to obtain the same results:

```
> attach(Baberuth)      # Assumes package PASWR is loaded
> NYYRBI <- RBI[7:21]   # Extract RBIs only while a NYY
> SNYYRBI <- sort(NYYRBI)
> p <- c(.25,.50,.75)
> n <- length(NYYRBI)
> n
[1] 15
> order.stat <- p*(n-1)+1
> order.stat
[1]  4.5  8.0 11.5
> Q1 <- SNYYRBI[4]+.5*(SNYYRBI[5]-SNYYRBI[4])
> Q2 <- SNYYRBI[8]
> Q3 <- SNYYRBI[11]+.5*(SNYYRBI[12]-SNYYRBI[11])
> QU <- c(Q1, Q2, Q3)
> names(QU) <- c("Q1","Q2","Q3")
> QU
   Q1    Q2    Q3
112.0 137.0 153.5
```

```
> quantile(NYYRBI, probs=c(.25,.50,.75))
  25%   50%   75%
112.0 137.0 153.5
> j <- (floor((n+1)/2)+1)/2     # Number to count in
> j
[1] 4.5
> lower.hinge <- SNYYRBI[4]+.5*(SNYYRBI[5]-SNYYRBI[4])
> upper.hinge <- SNYYRBI[11]+.5*(SNYYRBI[12]-SNYYRBI[11])
> small <- min(NYYRBI)
> large <- max(NYYRBI)
> five.numbers <- c(small, lower.hinge, Q2, upper.hinge, large)
> five.numbers
[1]  66.0 112.0 137.0 153.5 171.0
> fivenum(NYYRBI)                   # Only works in R
[1]  66.0 112.0 137.0 153.5 171.0
> detach(Baberuth)
```

In this particular example, the first and third quartile are equal to the lower and upper hinge, respectively. ■

2.5.5 Boxplots

A popular method of representing the information in a five-number summary is the **boxplot**. To show spread, a box is drawn from the lower hinge (H_L) to the upper hinge (H_U) with a vertical line drawn through the box to indicate the median or second quartile (Q_2). A "whisker" is drawn from H_U to the largest data value that does not exceed the upper fence. This value is called the **adjacent value**. The upper fence is defined as $Fence_U = H_U + 1.5 \times H_{spread}$, where $H_{spread} = H_U - H_L$. A whisker is also drawn from H_L to the smallest value that is larger than the lower fence, where the lower fence is defined as $Fence_L = H_L - 1.5 \times H_{spread}$. Any value smaller than the lower fence or larger than the upper fence is considered an **outlier** and is generally depicted with a hollow circle. Figure 2.10 on the following page illustrates a boxplot for the variable `fat` from the data frame `Bodyfat`.

To create a boxplot with S, use the command `boxplot()`. By default, boxplots in R have a vertical orientation. To create a horizontal boxplot with R, use the optional argument `horizontal=TRUE`. Currently, S-PLUS does not have an option to produce horizontal boxplots with the `boxplot()` function. However, S-PLUS does have the function `bwplot()`, which produces horizontal boxplots. Common arguments for `boxplot()` include `col=` to set the box color and `notch=TRUE` to add a notch to the box to highlight the median.

Example 2.14 Use the data frame `Cars93` in the `MASS` package to create a boxplot of the variable `Min.Price`. Use the `text()` function to label the five-number summary values in the boxplot.

Solution: Two solutions are presented: one for R and one for S-PLUS. The solution for S-PLUS is slightly more involved because S-PLUS does not have a built-in function to compute the five-number summary. The final boxplot from R is shown in Figure 2.11 on page 47. Additionally, the labels in R contain mathematical notation. To learn more about R's ability to plot mathematical expressions, type `?mathplot` at the R prompt.

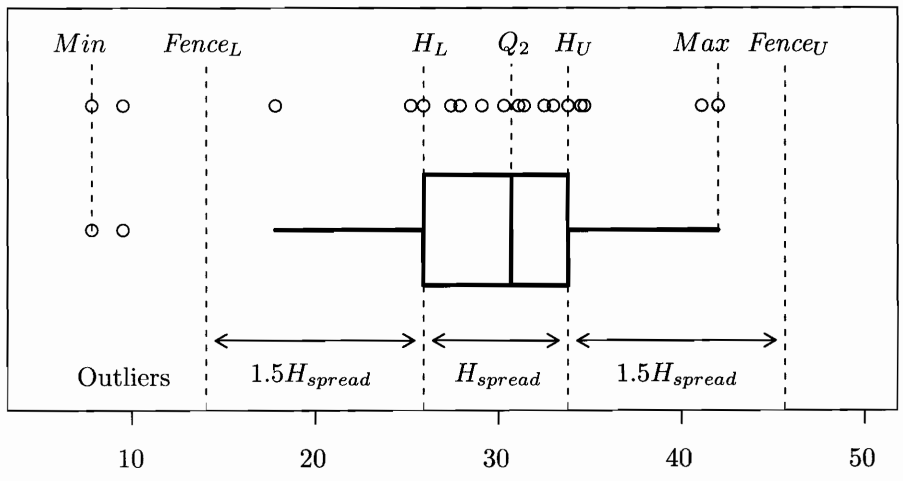

FIGURE 2.10: Graph depicting the five-number summary in relationship to original data and the boxplot

Solution for R:

```
> library(MASS)
> attach(Cars93)
> boxplot(Min.Price, ylab="Minimum Price (in $1000) for basic
+ version", col="gray")
> f <- fivenum(Min.Price)
> text(rep(1.25,5), f, labels=c("Min", expression(H[L]),
+ expression(Q[2]) , expression(H[U]), "Max"), pos=4)
> detach(Cars93)    # Clean up
```

Solution for S-PLUS:

```
> library(MASS)
> attach(Cars93)
> n <- length(Min.Price)
> smp <- sort(Min.Price)
> count <- (floor((n+1)/2)+1)/2  # Using Equation 2.4
> count
[1] 24
> lower.hinge <- smp[count]
> upper.hinge <- smp[(n-count+1)]
> five.num <- c(min(smp), lower.hinge, median(smp), upper.hinge,
+ max(smp))
> boxplot(Min.Price, ylab="Minimum Price (in $1000) for basic
+ version")
> text(rep(85,5), five.num, labels=c("Minimum", "Lower Hinge",
+ "Median", "Upper Hinge", "Maximum"))
> detach(Cars93)    # Clean up
```

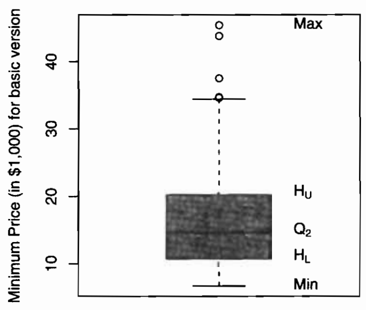

FIGURE 2.11: Boxplot of car prices with five-number summaries labeled

Boxplots are useful for detecting skewness, finding outliers, and comparing two or more variables that are all measured on the same scale. However, a boxplot will not detect multi-modality. ■

2.6 Summary Measures of Spread

Summary measures of center such as the mean and median are important because they describe "typical" values in a data set. However, it is possible to have two data sets with the same means (Example 2.11 on page 41) and/or medians while still having different spreads. For this reason, it is important to measure not only typical values but also the spread of the values in a distribution in order to describe the distribution fully. There are many ways to measure spread, some of which include range, interquartile range, and variance.

2.6.1 Range

The easiest measure of spread to compute is the range. At times, the range refers to the difference between the smallest value in a data set and the largest value in the data set. Other times, the range refers to the smallest and largest values of a data set as a pair. The S function `range(x)` returns the smallest and largest values in `x`. If the distance between the largest and smallest value is desired, one can use `diff(range(x))`:

```
> range(1:10)
[1]  1 10
> diff(range(1:10))
[1] 9
```

2.6.2 Interquartile Range

Instead of looking at the entire range of the data, looking at the middle 50% will often prove to be a useful measure of spread, especially when the data are skewed. The interquartile range (IQR) is defined as $IQR = Q_3 - Q_1$ and can be found with the function `IQR()`:

```
> quantile(1:10)
   0%   25%   50%   75%  100%
 1.00  3.25  5.50  7.75 10.00
> IQR(1:10)
[1] 4.5
```

2.6.3 Variance

The **sample variance**, s^2, can be thought of as the average squared distance of the sample values from the sample mean. It is not quite the average because the quantity is divided by $n-1$ instead of n in the formula

$$s^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}. \tag{2.5}$$

When the positive square root of the sample variance is taken, the **sample standard deviation**, s, results. It is often preferable to report the sample standard deviation instead of the variance since the units of measurement for the sample standard deviation are the same as those of the individual data points in the sample. To compute the variance with S, use the function `var(x)`. One could compute the standard deviation by taking the square root of the variance `sqrt(var(x))` or use the built-in function to do so. However, be aware that the function to compute the standard deviation in R is `sd(x)`, while the function to compute the standard deviation in S-PLUS is `stdev(x)`. The standard deviation is an appropriate measure of spread for normal distributions:

```
> x <- 1:5
> x
[1] 1 2 3 4 5
> n <- length(x)
> mean.x <- mean(x)
> mean.x
[1] 3
> x-mean.x
[1] -2 -1  0  1  2
> (x-mean.x)^2
[1] 4 1 0 1 4
> NUM <- sum((x-mean.x)^2)  # numerator of s^2 hard way
> NUM
[1] 10
> DEN <- n-1                # denominator of s^2
> DEN
[1] 4
> VAR <- NUM/DEN            # variance hard way
> VAR
[1] 2.5
> var(x)                    # variance easy way
[1] 2.5
> SD <- sqrt(VAR)           # standard deviation hard way
> SD
[1] 1.581139
```

```
> sd(x)                    # standard deviation with R
[1] 1.581139
> stdev(x)                 # standard deviation with S-PLUS
[1] 1.581139
```

An interesting function that will return different results depending on the class of the object to which it is applied is the S function `summary()`. When the object is a numeric vector, as is the case with `x`, six summary statistics are returned: the minimum, the first quartile, the median, the mean, the third quartile, and the maximum:

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       2       3       3       4       5
```

2.7 Bivariate Data

Methods to summarize and display relationships between two variables (bivariate data) will be the focus of the next few pages. In Section 2.3, two of the methods used to gain a deeper understanding of categorical variables were tables and barplots. When two variables are categorical, tables, called contingency tables, and barcharts will still prove useful. When presented with quantitative bivariate data, relevant questions will deal with the relationships between the two variables. For example, is there a relationship between a person's height and his weight? Is there a relationship between the amount of time a student spends studying and her grades? Graphical techniques such as scatterplots can be used to explore bivariate relationships. When relationships exist between variables, different correlation coefficients are used to characterize the strengths of the relationships. Finally, a brief introduction to the simple linear regression model is given before moving into multivariate data.

2.7.1 Two-Way Contingency Tables

The S command `table(x)` was used for creating frequency tables with univariate, categorical variables. For bivariate, categorical data, the command `table(x, y)` is used to create two-way contingency tables where `x` and `y` represent the two categorical variables.

Example 2.15 Consider the data frame `EPIDURAL`, which contains information from a study to determine whether the traditional sitting position or the hamstring stretch position is superior for administering epidural anesthesia to pregnant women in labor as measured by the number of obstructive (needle to bone) contacts (OC). The variable `Doctor` specifies which of the four physicians in the study administered the procedure. `Ease` is the physician's assessment prior to administering the epidural of how well bony landmarks for epidural placement can be felt. Produce a two-way contingency table for the variables `Doctor` and `Ease`.

Solution: The goal is to produce a two-way table such as the one in Table 2.2 on the next page with S. The levels of categorical variables by default are alphabetical. Consequently, the levels of `Ease` are Difficult, Easy, and Impossible. Pay particular attention to how the levels of a variable can be rearranged in the code that follows.

Table 2.2: Two-way table of Doctor by Ease

	Easy	Difficult	Impossible
Dr. A	19	3	1
Dr. B	7	10	4
Dr. C	18	3	0
Dr. D	13	4	3

```
> head(EPIDURAL, n=5)    # Shows first five rows of EPIDURAL
  Doctor  kg  cm       Ease            Treatment OC Complications
1  Dr. B 116 172  Difficult Traditional Sitting  0          None
2  Dr. C  86 176       Easy   Hamstring Stretch  0          None
3  Dr. B  72 157  Difficult Traditional Sitting  0          None
4  Dr. B  63 169       Easy   Hamstring Stretch  2          None
5  Dr. B 114 163 Impossible Traditional Sitting  0          None
> attach(EPIDURAL)
> table(Doctor, Ease)    # Levels of Ease not in increasing order
       Ease
Doctor  Difficult Easy Impossible
  Dr. A         3   19          1
  Dr. B        10    7          4
  Dr. C         3   18          0
  Dr. D         4   13          3
> Teasy <- factor(Ease, levels=c("Easy","Difficult","Impossible"))
> table(Doctor, Teasy)   # Levels of Ease in increasing order
       Teasy
Doctor  Easy Difficult Impossible
  Dr. A   19         3          1
  Dr. B    7        10          4
  Dr. C   18         3          0
  Dr. D   13         4          3
```

Although the command `table(Doctor, Ease)` produced a two way contingency table, reordering the levels of `Ease` produces a more readable two-way contingency table. ■

Extensions to muti-way contingency tables can be accomplished by specifying additional factors to the `table()` function or by using the R flattened table function `ftable()`. A flattened three-way contingency table of the factors `Doctor`, `Treatment`, and `Teasy` follows. More options for both `table` and `ftable` can be found in their respective help files.

```
> ftable(Doctor, Treatment, Teasy)
                           Teasy Easy Difficult Impossible
Doctor Treatment
Dr. A  Hamstring Stretch            7         1          0
       Traditional Sitting         12         2          1
Dr. B  Hamstring Stretch            3         3          0
       Traditional Sitting          4         7          4
Dr. C  Hamstring Stretch            8         3          0
       Traditional Sitting         10         0          0
Dr. D  Hamstring Stretch            7         1          2
       Traditional Sitting          6         3          1
> detach(EPIDURAL)
```

2.7.2 Graphical Representations of Two-Way Contingency Tables

Barplots can be used to depict graphically the information from two-way contingency tables. This is accomplished by picking one of the variables to form the categories of the barplot. Next, the second variable's levels are graphed either in a single bar (stacked) or as several bars (side-by-side).

Example 2.16 Produce stacked and side-by-side barplots of the information contained in Table 2.2 on the facing page.

Solution: Barplots where the variable of interest is `Ease` then `Doctor` are created first. Subsequently, side-by-side barplots where the variables of interest are `Ease` then `Doctor` are created. The graphs in Figure 2.12 on the next page were created using R. Output from S-PLUS will look slightly different. The user should consult the on-line documentation using `?barplots` for the differences between R and S-PLUS.

```
> attach(EPIDURAL)
> Teasy <- factor(Ease, levels=c("Easy","Difficult","Impossible"))
> X <- table(Doctor, Teasy)
> X

       Teasy
Doctor  Easy Difficult Impossible
  Dr. A   19         3          1
  Dr. B    7        10          4
  Dr. C   18         3          0
  Dr. D   13         4          3
> t(X)      # Transpose X
            Doctor
Teasy        Dr. A Dr. B Dr. C Dr. D
  Easy          19     7    18    13
  Difficult      3    10     3     4
  Impossible     1     4     0     3

> par(mfrow=c(2,2))
> barplot(X, main="Barplot where Doctor is Stacked \n within Levels
+ of Palpitation")
> barplot(t(X), main="Barplot where Levels of Palpitation \n is
+ Stacked within Doctor")
> barplot(X, beside=TRUE, main="Barplot where Doctor is Grouped \n
+ within Levels of Palpitation")
> barplot(t(X), beside=TRUE, main="Barplot where Levels of Palpitation
+ \n is Grouped within Doctor")
> par(mfrow=c(1,1))
> detach(EPIDURAL)
```

From the example, it is seen that the categories for the barplot are the numeric columns in a two-way contingency table. If the user wants the categories to be reversed, transpose the table using the command `t(table(x, y))`, where `table(x, y)` is the two-way contingency table. ■

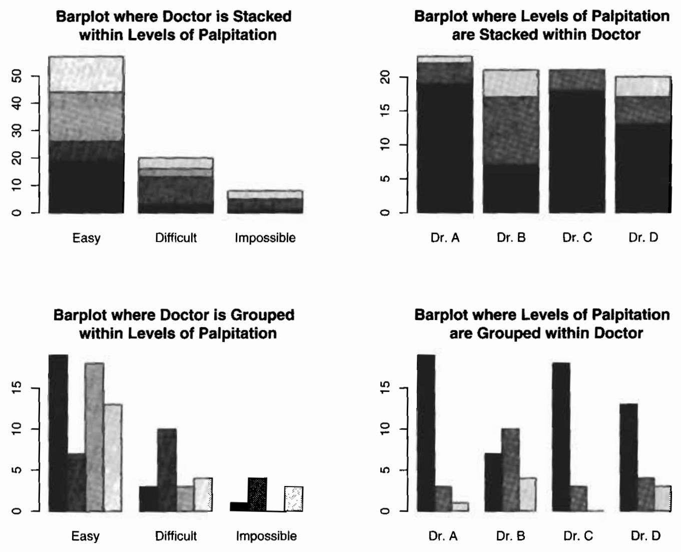

FIGURE 2.12: Stacked and side-by-side barplots for levels of palpitation (Teasy) and physician (Doctor)

Relationships are often better represented with proportions than with counts. R has the function `prop.table(x)`, which can be used to compute proportions based on the number of entries in either the entire table, `x`, which is the default, or by entering `prop.table(x, margin=1)` for row totals or `prop.table(x, margin=2)` for column totals.

Example 2.17 Using the data frame `EPIDURAL`, create a side-by-side barplot of `Treatment` versus `OC`.

Solution: Since there have been 25 patients treated with the hamstring stretch position and 49 patients treated with the traditional sitting position, it would not be rational to compare the frequencies. Instead, one should compare the percentages within the categories of `OC` by `Treatment`:

```
> attach(EPIDURAL)
> table(Treatment, OC)
                      OC
Treatment              0  1  2  3  4  5  6 10
  Hamstring Stretch   17  6  6  2  1  1  0  2
  Traditional Sitting 23 16  3  1  2  2  2  0
> addmargins(table(Treatment, OC)) # addmargins is an R command
                      OC
Treatment              0  1 2 3 4 5 6 10 Sum
  Hamstring Stretch   17  6 6 2 1 1 0  2  35
  Traditional Sitting 23 16 3 1 2 2 2  0  49
  Sum                 40 22 9 3 3 3 2  2  84
> X <-prop.table(table(Treatment, OC),1)  # Percents by rows
> X
                     OC
Treatment                      0          1          2          3
  Hamstring Stretch   0.48571429 0.17142857 0.17142857 0.05714286
  Traditional Sitting 0.46938776 0.32653061 0.06122449 0.02040816
                     OC
Treatment                      4          5          6         10
  Hamstring Stretch   0.02857143 0.02857143 0.00000000 0.05714286
  Traditional Sitting 0.04081633 0.04081633 0.04081633 0.00000000
```

```
> par(mfrow=c(2,1))
> barplot(X, beside=TRUE, legend=TRUE)
> barplot(t(X), beside=TRUE, legend=TRUE)
> par(mfrow=c(1,1))
> detach(EPIDURAL)
```

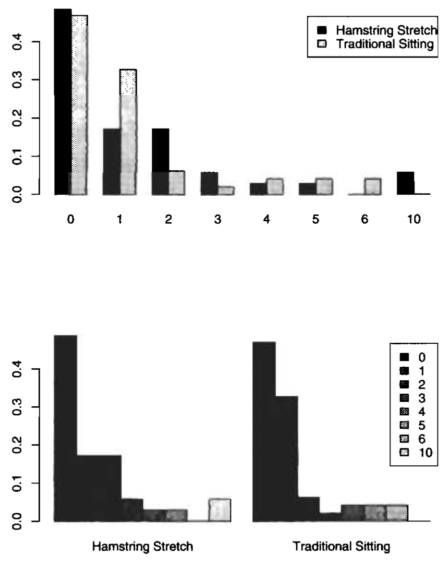

FIGURE 2.13: Side-by-side barplots showing percentages in obstructive contacts categories by treatments

Note that the categories for the barplot in the upper graph of Figure 2.13 are the OC categories in the two-way contingency table. Within each OC category, comparisons are shown side-by-side based on the treatment. If the user wants the categories to be reversed, transpose the table using the command `t(table(x, y))`, where `table(x, y)` is the two-way contingency table. ■

2.7.3 Comparing Samples

The need to compare two samples is quite common. Simple experiments will often compare a control group to a treatment group in an effort to see if the treatment provides some added benefit. For example, the data in the `EPIDURAL` data frame are from an ongoing experiment to see which of two positions results in fewer obstructive bone contacts (the times the needle hits a bone). When comparing two samples, typically some type of inference to the samples' populations is desired. That is, are the centers the same? Are the spreads similar? Are the shapes of the two distributions similar? Graphs such as histograms, density plots, boxplots, and quantile-quantile plots can help answer these questions. Histograms and density plots were introduced in Section 2.4, and boxplots were introduced in Section 2.5. A **quantile-quantile (Q-Q) plot** plots the quantiles of one distribution against the quantiles of another distribution as (x, y) points. When the two distributions have similar shapes, the points will fall along a straight line. The S function to make a quantile-quantile plot is `qqplot(x, y)`. Histograms can be used to compare two distributions. However, it is rather challenging to put both histograms on the same graph. Example 2.18 shows the user how histograms can be used to compare distributions. However, a better approach is to use Trellis/lattice graphics, which are explained in Section 2.8.

Example 2.18 Use histograms to compare the body weight index (BWI) for the two treatments (traditional sitting and hamstring stretch stored in `Treatment`) using the data frame `EPIDURAL`.

Solution: First, BWI is typically defined as kg/m^2. Since the data frame `EPIDURAL` does not contain a BWI variable, one is created. Subsequently, the default options for the BWI histograms of the control and treatment groups are shown in the first column of Figure 2.14 on the next page, while the BWI histograms of the control and treatment groups are shown in the second column of Figure 2.14 after the axes limits for both the x- and y-axes have been set to the same values for both histograms:

```
> attach(EPIDURAL)
> BWI <- kg/(cm/100)^2
> Control <- BWI[Treatment=="Traditional Sitting"]
> Treated <- BWI[Treatment=="Hamstring Stretch"]
> par(mfrow=c(2,2))      # 2*2 plotting region
> hist(Control)
> hist(Control, xlim=c(20,60), ylim=c(0,17))
> hist(Treated)
> hist(Treated, xlim=c(20,60), ylim=c(0,17))
> par(mfrow=c(1,1))      # 1*1 plotting region
> detach(EPIDURAL)
```

Note that it is misleading to compare histograms where the bin widths and/or units on the axes of the two histograms are different. Note that both axes are different in the first column of Figure 2.14 on the facing page. The bins of the two histograms are set with the argument `breaks=`, and the x- and y-axes are set with the arguments `xlim=` and `ylim=`, respectively. The general shape of the BWI for the patients administered epidurals in the hamstring stretch position is unimodal skewed to the right. While the distribution of BWI for patients administered epidurals in the traditional sitting position is also unimodal skewed to the right, it is not quite as skewed as the distribution where patients are administered epidurals from the hamstring stretch position. ■

Example 2.19 Use side-by-side boxplots and superimposed density plots to compare the BWI for the two treatments (traditional sitting and hamstring stretch stored in `Treatment`) using the data frame `EPIDURAL`.

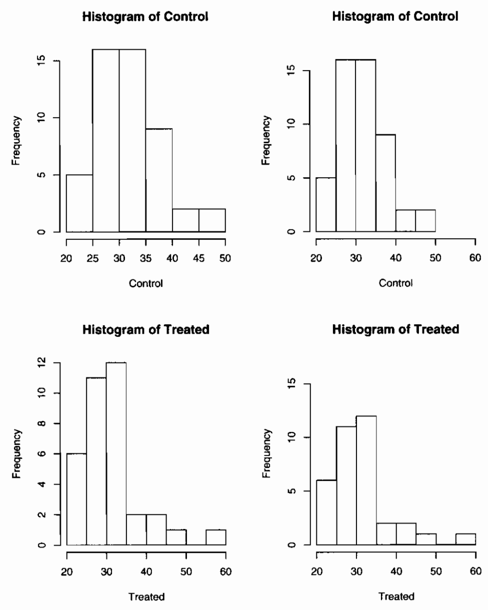

FIGURE 2.14: Side-by-side barplots showing percentages in obstructive contacts' categories by treatments

Solution: The argument `horizontal=TRUE` used in the `boxplot()` function will only work in R. One way to create horizontal boxplots with S-PLUS is to use the Trellis function `bwplot()`. Specifically, one might enter `bwplot(Treatment~BWI)` to produce side-by-side boxplots with S-PLUS. Trellis/lattice graphs will be discussed in more detail in Section 2.8. Using boxplots, as seen in Figure 2.15 on the next page, one sees that the median for both treatments is around 30 kg/m^2 and both distributions appear to be skewed to the right.

```
> attach(EPIDURAL)
> par(pty="s")          # Make plotting region square
> BWI <- kg/(cm/100)^2  # Define body weight index
> Control <- BWI[Treatment=="Traditional Sitting"]
> Treated <- BWI[Treatment=="Hamstring Stretch"]
> boxplot(Control, Treated, horizontal=TRUE, col=c(13,4),
+ names=c("Traditional Sitting","Hamstring Stretch"), las=1)
> plot(density(Control), xlim=c(20,60), col=13, lwd=2, main="", xlab="")
> lines(density(Treated), lty=2, col=4, lwd=2)
> detach(EPIDURAL)
```

The density plot in Figure 2.16 on the following page further indicates that the distributions for the BWI for both the traditional sitting and the hamstring stretch position are skewed to the right. ■

Example 2.20 Use a quantile-quantile plot to compare the BWI for the two treatments (traditional sitting and hamstring stretch stored in `Treatment`) using the data frame `EPIDURAL`.

Solution: Commands to recreate the quantile-quantile plot shown in Figure 2.17 on page 57 follow. Note that both the x- and y-axes have the same limits.

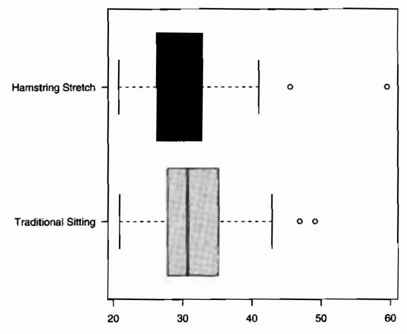

FIGURE 2.15: Side-by-side boxplots of BWI in the traditional sitting and hamstring stretch positions

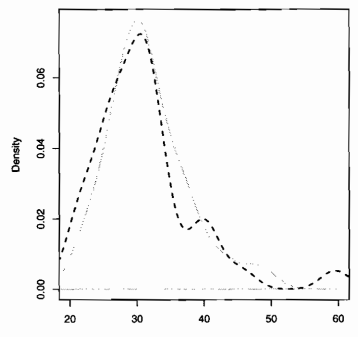

FIGURE 2.16: Density plots of BWI in the traditional sitting (solid line) and hamstring stretch positions (dashed line)

```
> attach(EPIDURAL)
> par(pty="s")          # Make plotting region square
> qqplot(Control, Treated, xlim=c(20,60), ylim=c(20,60))
> abline(a=0, b=1)      # Line y=0+1*x
> par(pty="m")          # Maximize plotting region
> detach(EPIDURAL)
```

The quantile-quantile plot in Figure 2.17 suggests the distributions are fairly similar since the points roughly follow the $y = x$ line. ■

2.7.4 Relationships between Two Numeric Variables

Relationships between two numeric variables can be viewed with **scatterplots**. A scatterplot plots the values of one variable against the values of a second variable as points (x_i, y_i) in the Cartesian plane. Typical questions researchers seek to answer with numeric variables include "Is there a relationship between the two variables?", "How strong is the relationship between the two variables?", and "Is the relationship linear?" Questions such as

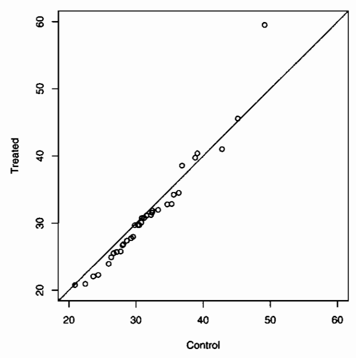

FIGURE 2.17: Quantile-quantile plot of BWI in the traditional sitting and hamstring stretch positions

"Is there a relationship between a person's height and his weight?" or "Is there a relationship between a student's grades and the time spent studying?" are typical. Given two numeric variables, say x and y, entering the S function `plot(x, y)` produces a scatterplot.

Example 2.21 Use the data frame `Animals` from the `MASS` package to investigate whether the brain weights of animals are related to their body weights. In other words, is a bigger brain required to govern a bigger body?

Solution: Because of the large range in body and brain weights, (0.023 kg to 87,000 kg) and (0.4 g to 5712 g), respectively, a scatterplot of the values in `body` and `brain` is too distorted to reveal any clear pattern. Consequently, the data is transformed by taking natural logarithms of both variables and plotting the resulting values as shown in Figure 2.18 on the following page.

```
> library(MASS)
> attach(Animals)
> range(body)
[1] 2.3e-02 8.7e+04
> range(brain)
[1]    0.4 5712.0
> range(log(body))
[1] -3.772261 11.373663
> range(log(brain))
[1] -0.9162907  8.6503245
> par(pty="s")
> plot(log(body), log(brain))
> identify(log(body), log(brain), labels=row.names(Animals))
> detach(Animals)
```

The function `identify()` was used to label several of the points in Figure 2.18 on the next page. The function `identify()` labels the closest point in the scatterplot with each mouse click (left click with windows) until instructed to stop. How the function is instructed to stop varies by operating system. Right clicking with windows, middle clicking with Linux, and using the escape key in Mac OS X will generally stop the identification process. Based on Figure 2.18, there appears to be linear relationship between the logarithm of the body weights and the logarithm of the brain weights. The dinosaurs can be classified as bivariate outliers as they do not fit the overall pattern seen in the rest of the data. ■

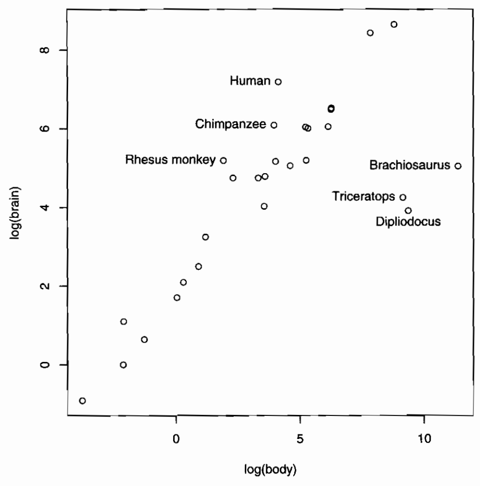

FIGURE 2.18: Scatterplot of `log(brain)` versus `log(body)` for Example 2.21

2.7.5 Correlation

The **correlation coefficient**, denoted by r, measures the strength and direction of the linear relationship between two numeric variables X and Y and is defined by

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_X}\right)\left(\frac{y_i - \bar{y}}{s_Y}\right) \tag{2.6}$$

The value for r will always be between -1 and $+1$. When r is close to $+1$, it indicates a strong positive linear relationship. That is, when x increases so does y, and vice versa. When the value of r is close to -1, it indicates a strong negative linear relationship. Values of r close to zero indicate weak linear relationships. To compute the correlation between two numeric vectors with S, one may use the function `cor(x, y)`.

Example 2.22 Find the correlation coefficient, r, between the logarithms of the body and brain weights in the data frame `Animals` from the `MASS` package using (2.6). Verify the calculated answer using the S function `cor()`.

Solution: First, the variables `logbody`, `logbrain`, `Zbody`, and `Zbrain` are created. The new variables are subsequently column binded to the `Animals` data frame and stored in a new data frame named `Anim`. Note that the output uses the R function `sd()` to compute the standard deviation. To compute the standard deviation with S-PLUS, use `stdev()`.

```
> attach(Animals)
> options(digits=3)          # Three digits for output
> logbody <- log(body)
> logbrain <- log(brain)
> Zbody <- (logbody - mean(logbody))/sd(logbody)
> Zbrain <- (logbrain - mean(logbrain))/sd(logbrain)
> Anim <- cbind(Animals, logbody, logbrain, Zbody, Zbrain)
> n <- length(logbody)
> r <- (1/(n-1))*sum(Zbody*Zbrain)  # Definition of r
> r
[1] 0.78
> cor(logbody, logbrain)
[1] 0.78
> Anim[1:6,]                # Show first 6 rows of Anim
                     body  brain logbody logbrain   Zbody  Zbrain
Mountain beaver  1.35e+00    8.1  0.3001    2.092 -0.9206 -0.9726
Cow              4.65e+02  423.0  6.1420    6.047  0.6287  0.6760
Grey wolf        3.63e+01  119.5  3.5926    4.783 -0.0474  0.1492
Goat             2.77e+01  115.0  3.3200    4.745 -0.1197  0.1332
Guinea pig       1.04e+00    5.5  0.0392    1.705 -0.9898 -1.1340
Dipliodocus      1.17e+04   50.0  9.3673    3.912  1.4841 -0.2140
```

The correlation between `logbrain` and `logbody` is 0.78, which indicates a positive linear relationship between the two variables. An alternative to computing the z-scores directly is to use the function `scale()`:

```
> ZBO <- scale(logbody)     # Z score of logbody
> ZBR <- scale(logbrain)    # Z score of logbrain
> SAME <- cbind(Zbody, ZBO, Zbrain, ZBR)
> SAME[1:5,]                # Show first five rows of data frame
       Zbody         Zbrain
[1,] -0.9206 -0.9206 -0.973 -0.973
[2,]  0.6287  0.6287  0.676  0.676
[3,] -0.0474 -0.0474  0.149  0.149
[4,] -0.1197 -0.1197  0.133  0.133
[5,] -0.9898 -0.9898 -1.134 -1.134
> detach(Animals)
```

■

2.7.6 Sorting a Data Frame by One or More of Its Columns

The `sort()` function can be used to sort a single variable in either increasing or decreasing order. However, if the user wants to sort a variable in a data frame and have the other variables reflect the new ordering, `sort()` will not work. The function needed to rearrange the values in a data frame to reflect the order of a particular variable or variables in the event of ties is `order()`. Given three variables `x`, `y`, and `z` in a data frame `DF`, the command `order(x)` returns the indices of the sorted values of `x`. Consequently, the data frame `DF` can be sorted by `x` with the command `DF[order(x),]`. In the event of ties, further arguments to `order` can be used to specify how the ties should be broken. Consider how ties are broken with the following numbers. To conserve space, the transpose function `t()` is used on the data frame `DF`.

```
> x <- c(1,1,1,3,3,3,2,2,2)
> y <- c(3,2,3,6,2,6,10,4,4)
> z <- c(7,4,2,9,6,4,5,3,1)
> DF <- data.frame(x, y, z)
> rm(x, y, z)
> attach(DF)
> t(DF)
  1 2 3 4 5 6  7 8 9
x 1 1 1 3 3 3  2 2 2
y 3 2 3 6 2 6 10 4 4
z 7 4 2 9 6 4  5 3 1
> t(DF[order(x, y, z),])
  2 3 1 9 8  7 5 6 4
x 1 1 1 2 2  2 3 3 3
y 2 3 3 4 4 10 2 6 6
z 4 2 7 1 3  5 6 4 9
> detach(DF)
```

Example 2.23 Find the correlation coefficient, r, between the logarithms of the body and brain weights in the data frame `Animals` from the `MASS` package with and without dinosaurs.

Solution: To save space, only four rows of the data frames `SA` and `NoDINO` are shown in the output. Note that there are a total of 28 animals in the data frame `Animals`.

```
> attach(Animals)
> cor(log(body), log(brain))
[1] 0.7794935
> SA <- Animals[order(body),]    # Sorted by body weight
> detach(Animals)
> tail(SA, n=4) # Equivalently SA[25:28,], shows four heaviest animals
                  body  brain
African elephant  6654 5712.0
Triceratops       9400   70.0
Dipliodocus      11700   50.0
Brachiosaurus    87000  154.5
> NoDINO <- SA[-(28:26),] # Remove rows 26-28 of SA
> attach(NoDINO)                 # NoDINO contains 25 rows
> NoDINO[22:25,]                 # Show four heaviest animals
                 body brain
Horse             521   655
Giraffe           529   680
Asian elephant   2547  4603
African elephant 6654  5712
> cor(log(body), log(brain))      # Correlation without dinosaurs
[1] 0.9600516
> detach(NoDINO)
```

The correlation between `log(brain)` and `log(body)` when dinosaurs are included is 0.78 and the correlation between `log(brain)` and `log(body)` is 0.96 when the dinosaurs are removed from the computation. ■

2.7.7 Fitting Lines to Bivariate Data

When a linear pattern is evident from a scatterplot, the relationship between the two variables is often modeled with a straight line. When modeling a bivariate relationship, Y is called the **response** or **dependent** variable, and x is called the **predictor** or **independent** variable. There are relationships that are of interest that are not linear. However, before addressing more complicated models, this material attempts to provide a foundation for the simpler models (simple linear regression) from which more complicated models can later be built. Chapter 12 is devoted to standard regression techniques for both the simple and multiple linear regression model. The simple linear regression model is written

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{2.7}$$

Model (2.7) is said to be simple, linear in the parameters (β_0 and β_1), and linear in the predictor variable (x_i). It is simple because there is only one predictor; linear in the parameters because no parameter appears as an exponent nor is multiplied or divided by another parameter; and linear in the predictor variable since the predictor variable is raised only to the first power. When the predictor variable is raised to a power, this power is called the **order** of the model. For now, only the simple linear model will be discussed. The goal is to estimate the coefficients β_0 and β_1 in (2.7). The most well-known method of estimating the coefficients β_0 and β_1 is to use ordinary least squares (OLS). OLS provides estimates of β_0 and β_1 by minimizing the sum of the squared deviations of the Y_is for all possible lines. Specifically, the sum of the squared residuals ($\hat{\varepsilon}_i = Y_i - \hat{Y}_i$) is minimized when the OLS estimators of β_0 and β_1 are

$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \bar{x} \tag{2.8}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\left(Y_i - \overline{Y}\right)}{\sum_{i=1}^{n} (x_i - \bar{x})^2}, \tag{2.9}$$

respectively. Note that the estimated regression function is written as

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i.$$

A graphical representation of the residuals and a line fit to some data using OLS can be seen in Figure 2.19 on the following page.

The OLS estimators of β_0 and β_1 are affected by outliers just as the mean and standard deviation are subject to outliers. Recall that the median and IQR were suggested as measures of center and spread, respectively, when working with skewed distributions. This recommendation was made because the median and IQR provide more robust measures of center and spread in the presence of outliers. In the presence of bivariate outliers, several robust alternatives exist for computing estimates of β_0 and β_1. Two alternatives to OLS implemented in the `MASS` package will be considered. Specifically, least-trimmed squares using the function `lqs()` and robust regression using an M estimator with the function `rlm()` are discussed. Just as OLS sought to minimize the squared vertical distance between all of the Y_is over all possible lines, least-trimmed squares minimizes the q smallest residuals over all possible lines where $q = \lfloor (n+p+1)/2 \rfloor$. Fitting for the function `rlm()` is done by iterated re-weighted least squares. Although `lqs()` and `rlm()` are computationally intensive, the interfaces for `lm()`, `lqs()`, and `rlm()` are essentially identical. All three functions require a model formula of the form `y~x`. The $\sim$ in this notation is read "is modeled by."

Example 2.24 In Exercise 2.23 on the preceding page, the correlation between the logarithms of the body and brain weights in the data frame `Animals` from the `MASS` package with and without dinosaurs was computed. Find the estimates for the least squares regression lines with and without dinosaurs where the logarithm of brain is modeled by the logarithm of body using Equations (2.8) and (2.9) as well as the S function `lm()`. Superimpose both lines on the scatterplot using the function `abline()` (see Table A.12 on page 667).

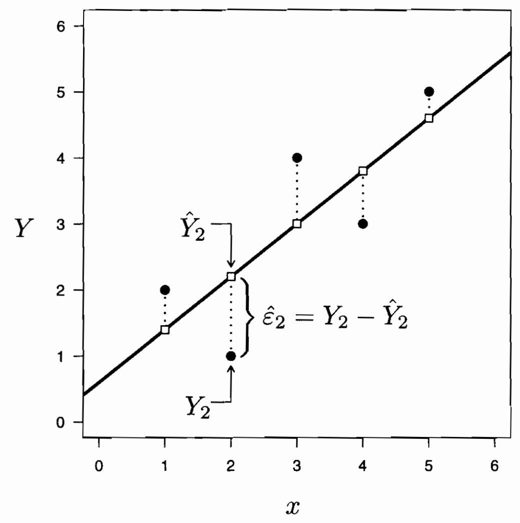

FIGURE 2.19: Graph depicting residuals. The vertical distances shown with a dotted line between the Y_is, depicted with a solid circle, and the $\hat{Y}_i$s, depicted with a clear square, are the residuals.

Solution: Recall that there are a total of 28 animals in the data frame Animals and 25 animals in the NoDINO data frame. The scatterplot with superimposed regression lines including the dinosaurs and omitting the dinosaurs is shown in Figure 2.20 on the next page.

```
> attach(Animals)
> Y <- log(brain)
> X <- log(body)
> plot(X, Y, xlab="log(body)", ylab="log(brain)")
> b1 <- sum((X-mean(X))*(Y-mean(Y)))/sum((X-mean(X))^2)
> b0 <- mean(Y) - b1*mean(X)
> estimates <- c(b0, b1)
> estimates
[1] 2.5548981 0.4959947
> modDINO <- lm(Y~X)
> modDINO

Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)            X
      2.555        0.496

> abline(modDINO, col="pink", lwd=2)
> SA <- Animals[order(body),]      # Sorted by body weight
> NoDINO <- SA[-(28:26),]          # Remove rows 26-28 (dinosuars)
> detach(Animals)
> attach(NoDINO)                    # NoDINO contains 25 rows
> Y <- log(brain)
> X <- log(body)
> b1 <- sum((X-mean(X))*(Y-mean(Y)))/sum((X-mean(X))^2)
> b0 <- mean(Y) - b1*mean(X)
> estimates <- c(b0, b1)
> estimates
[1] 2.1504121 0.7522607
> modNODINO <- lm(Y~X)
> modNODINO

Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)            X
     2.1504       0.7523

> abline(modNODINO, col="blue", lwd=2, lty=2)
> leglabels <- c("OLS with Dinosaurs", "OLS without Dinosaurs")
> leglty <- c(1,2)
> legcol=c("pink","blue")
> legend("topleft", legend=leglabels, lty=leglty, col=legcol, lwd=2)
> detach(NoDINO)
```

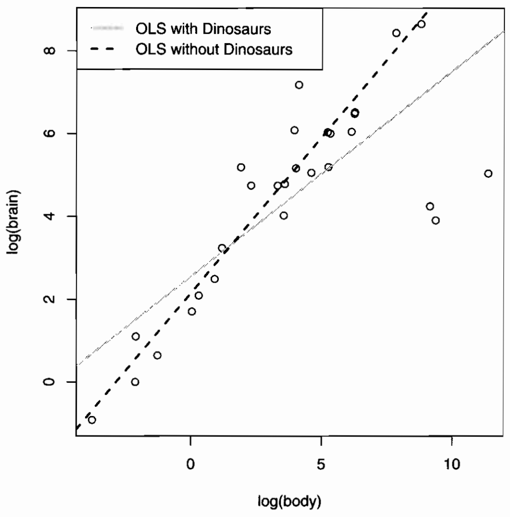

FIGURE 2.20: Scatterplot of `log(brain)` versus `log(body)` with superimposed regression lines computed with (solid line) and without (dashed line) dinosaurs

The intercept and slope of the regression line with dinosaurs are 2.555 and 0.496, respectively. Without the dinosaurs, the intercept and slope of the regression line are 2.1504 and 0.7523, respectively. ■

Example 2.25 From Figure 2.20 in Exercise 2.24 one notices three bivariate outliers (dinosaurs). Fit regression lines to the same data used in Exercise 2.20 using ordinary least squares, least-trimmed squares, and robust regression with an M estimator. Superimpose the resulting regression lines on a scatterplot and label the lines accordingly.

Solution: The scatterplot with the three superimposed regression lines is shown in Figure 2.21 on the facing page.

```
> attach(Animals)
> plot(log(body), log(brain), col="blue")
> f <- log(brain)~log(body)
> modelLM <- lm(f)
> modelLM
Call: lm(formula = f)

Coefficients:
(Intercept)    log(body)
      2.555        0.496

> abline(modelLM, col="pink", lwd=2)
> modelLQS <- lqs(f)
> modelLQS
Call: lqs.formula(formula = f)

Coefficients:
(Intercept)    log(body)
      1.816        0.776

Scale estimates 0.4637 0.4633

> abline(modelLQS, lty=2, col="red", lwd=2)
> modelRLM <- rlm(f, method="MM")
> modelRLM
Call: rlm(formula = f, method = "MM") Converged in 5 iterations

Coefficients:
(Intercept)   log(body)
  2.0486717   0.7512927

Degrees of freedom: 28 total; 26 residual
Scale estimate: 0.633

> abline(modelRLM, lty=3, col="black", lwd=2)
> leglabels <- c("Least Squares Line","Least-Trimmed Squares",
+ "Robust line: M-estimator ")
> leglty <- c(1,2,3)
> legend("topleft", legend=leglabels, lty=leglty,
+ col=c("pink","red","black"), lwd=2, cex=0.85)
> detach(Animals)
```

The least-trimmed squares (`lqs()`) procedure and the robust line with M estimator (`rlm()`) method produce lines that put relatively little importance on outliers (dinosaurs). This is

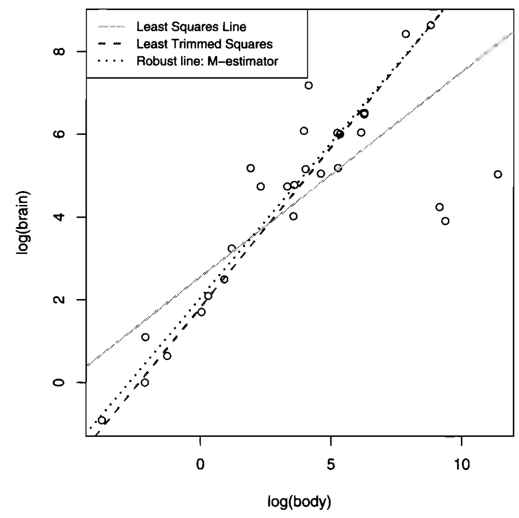

FIGURE 2.21: Scatterplot of `log(brain)` versus `log(body)` with three superimposed regression lines. Solid is the OLS line; dashed is the least-trimmed squares line; and dotted is the robust line.

further highlighted when one considers the estimates β_0 and β_1 for the OLS estimates without dinosaurs compared to the estimates of β_0 and β_1 for the least-trimmed squares and robust procedures given in Table 2.3.

Table 2.3: Different values for b_0 and b_1 with various regression methods

Method	b_0	b_1
OLS with dinosuars	2.555	0.496
OLS without dinosaurs	2.150	0.752
least-trimmed squares	1.816	0.776
robust line with M estimator	2.049	0.751

■

2.8 Multivariate Data (Lattice and Trellis Graphs)

This section examines tools that can be used to understand multivariate data. Specifically, Trellis displays (used in S-PLUS), which were developed by Cleveland (1993), are introduced. The R version of Cleveland's Trellis displays is implemented with the package `lattice`. Trellis displays are graphs that examine higher dimensional structure in data by conditioning on one or more variables. Trellis graphs are implemented in a slightly different fashion from traditional S graphs; however, some readers may find the layout, rendering, and default coloring of Trellis graphs more appealing than traditional S graphs. Trellis graphs are created with a formula syntax. The formula expresses the dependencies between the variables as follows:

`response ~ predictor | conditioning.variable`

The expression `y ~ x | z` is read "y is modeled as x given z." Depending on the type of graph, all three components may not need to be specified. Table A.11 on page 666 lists the arguments for some of the more popular Trellis functions. If there is more than one conditioning variable, they are all listed separated by the multiplication symbol (`*`).

Example 2.26 Use Trellis histograms to compare the body weight index (BWI) for the two treatments (traditional sitting and hamstring stretch stored in `Treatment`) using the data frame `EPIDURAL`.

Solution: Recall that BWI is typically defined as kg/m^2. Since the data frame `EPIDURAL` does not contain a BWI variable, one is created:

```
> attach(EPIDURAL)
> BWI <- kg/(cm/100)^2
> library(lattice)                          # only for R
> histogram(~BWI|Treatment, layout=c(1,2))
> detach(EPIDURAL)
```

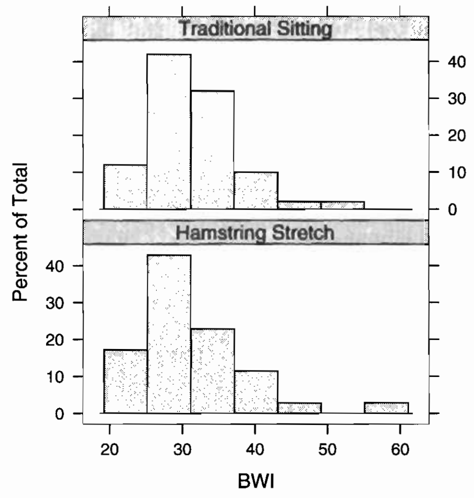

FIGURE 2.22: Comparative histograms of BWI by treament

The `histogram()` function used the additional argument of `layout=c(1,2)`. The first value of `layout` determines the number of columns (1) in the Trellis graph and the second value determines the number of rows (2) in the Trellis graph. This is in contrast to how dimensions are specified in a matrix, which is number of rows by number of columns. The basic shapes of the two histograms shown in Figure 2.22 are quite similar, just as was observed in Example 2.18 on page 54 when the histograms were created using traditional S graphs. ■

Example 2.27 In Example 2.19 on page 54 side-by-side boxplots were used to compare the BWI for the two treatments. An additional concern is that not only should the distribution of BWI be similar for treatments, but it should also be similar for each physician. Use Trellis graphs to create side-by-side boxplots of BWI by treatments given `Doctor` using the data frame `EPIDURAL`.

Solution: The argument `as.table=TRUE` used in the `bwplot()` function orders the graphs the way one reads a book. The default arrangement of graphs is to start in the lower left and move to the upper right. This is done so that the graphs appear with the smallest values in the lower left, analogous to a scatterplot.

```
> attach(EPIDURAL)
> BWI <- kg/(cm/100)^2
> library(lattice)
> bwplot(Treatment~BWI|Doctor, as.table=TRUE)  # Order: as one reads
```

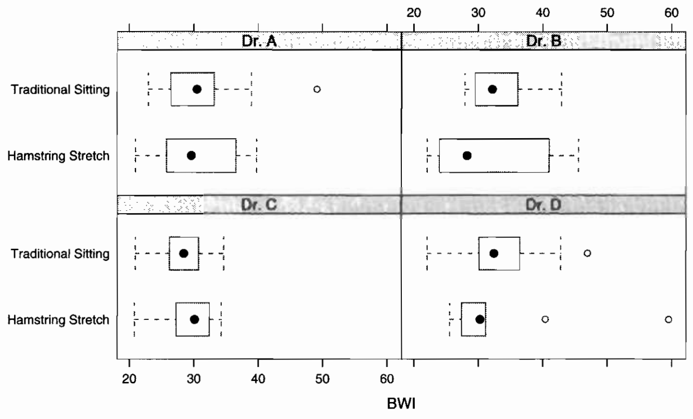

FIGURE 2.23: Trellis side-by-side boxplots of BWI in the traditional sitting and hamstring stretch positions given `Doctor`

Since the number of observations for each of the treatments is relatively small (range is from 6 to 15), it might be a better to look at the data with a **stripplot**. A stripplot of the treatments conditioning on physician is illustrated in Figure 2.24 on the following page.

```
> stripplot(Treatment~BWI|Doctor, jitter=TRUE, as.table=TRUE)
> detach(EPIDURAL)
```

The optional argument `jitter=TRUE` adds a small amount of noise to the values in the stripplot so that overlapping values are easier to distinguish. Based on the stripplots shown in Figure 2.24 on the next page, it seems that Dr. C's patients have a consistently smaller BWI for both treatment positions. Further investigation is needed to see why Dr. C's patients have consistently smaller BWI measurements versus the other physicians.

2.8.1 Arranging Several Graphs on a Single Page

The arrangement of Trellis graphs on a single page is again different from the arrangement of traditional graphs on a single page. Two different approaches can be taken when

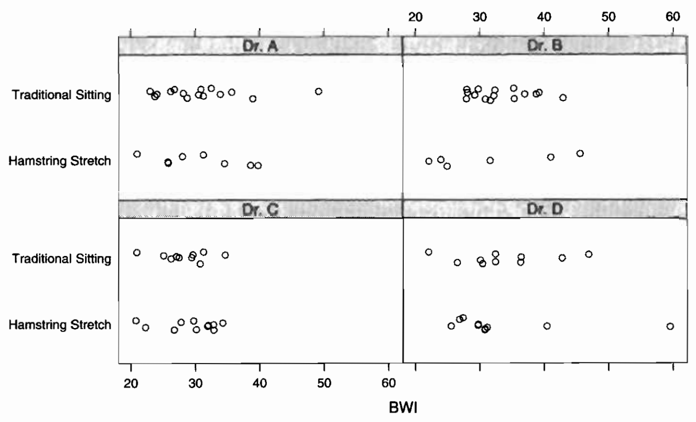

FIGURE 2.24: Trellis side-by-side stripplots of BWI in the traditional sitting and hamstring stretch positions given `Doctor`

arranging several graphs on a single page. The first approach discussed is to arrange the graphs in equally sized rectangles based on the dimensions of a matrix. In other words, if one wants to plot six graphs on a single page, it might be accomplished with a 3 by 2 or a 2 by 3 matrix where each position of the matrix represents a graph. To print each graph, the following structure is used:

```
print(trellisgraph, split=c(column, row, number of columns,
number of rows), more=TRUE/FALSE)
```

A second approach to producing multiple graphs on a single page is to literally specify the lower left and upper right coordinates for each graph. The lower left of the graph is denoted by the coordinates (0, 0), and the upper right corner is denoted by the coordinates (1, 1). The form for specifying each graph is $(x_{LL}, y_{LL}, x_{UR}, y_{UR})$. To print each graph, the following structure is used:

```
print(trellisgraph, position=c(x_LL, y_LL, x_UR, y_UR), more=TRUE/FALSE)
```

Example 2.28 Use Trellis graphs to create boxplots of BWI given `Doctor`, a scatterplot of `cm` versus `kg` given `Doctor`, a histogram of `BWI`, and a density plot of `BWI` given `Treatment` using the data frame `EPIDURAL`. Show all four graphs on the same page.

Solution: The solution provided is for R. The commands that follow will work in S-PLUS for graphs 2–4. However, the command `bwplot(~BWI|Doctor)` (graph 1) will not work in S-PLUS. The argument `as.table=TRUE` used in the `bwplot()` and the `xyplot()` functions are not requested in the problem. However, they are used since most people like to read from left to right and top to bottom. The four graphs are created and stored in variables named `graph1`, `graph2`, `graph3`, and `graph4`, respectively. By splitting the graph into a 2 by 2 matrix or by specifying the position for each of the four graphs one can reproduce Figure 2.25 on the facing page using the commands that follow.

```
> attach(EPIDURAL)
> library(lattice)
> graph1 <- bwplot(~BWI|Doctor, as.table=TRUE)
> graph2 <- xyplot(cm~kg|Doctor, as.table=TRUE)
> graph3 <- histogram(~BWI)
> graph4 <- densityplot(~BWI|Treatment)
> print(graph1, split=c(1,2,2,2), more=TRUE)  # Lower left
> print(graph2, split=c(2,2,2,2), more=TRUE)  # Lower right
> print(graph3, split=c(1,1,2,2), more=TRUE)  # Upper left
> print(graph4, split=c(2,1,2,2), more=FALSE) # Upper right
```

Using the literal position of the graph

```
> print(graph1, position=c(0,0,.5,.5), more=TRUE)  # Lower left
> print(graph2, position=c(.5,0,1,.5), more=TRUE)  # Lower right
> print(graph3, position=c(0,.5,.5,1), more=TRUE)  # Upper left
> print(graph4, position=c(.5,.5,1,1), more=FALSE) # Upper right
> detach(EPIDURAL)
```

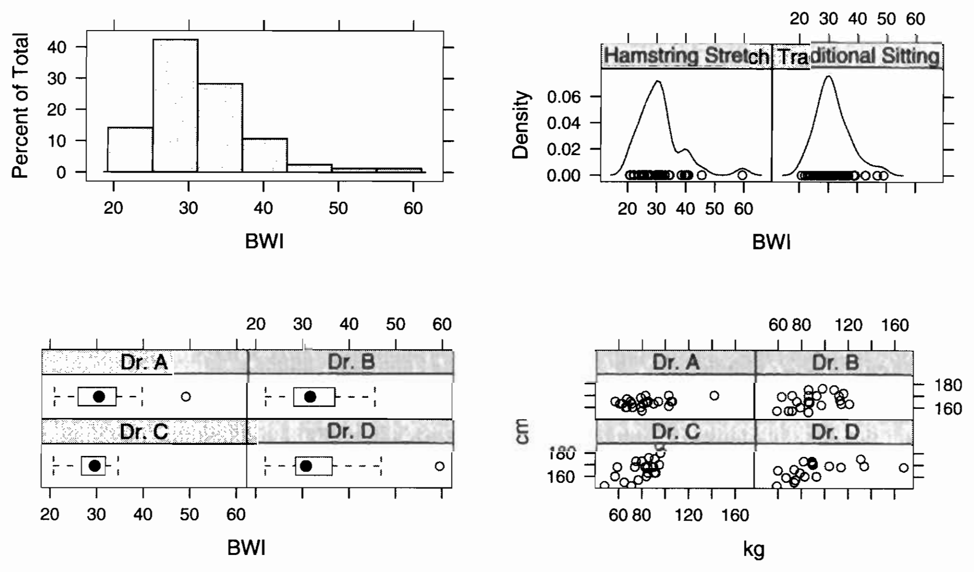

FIGURE 2.25: Arrangement of four different Trellis graphs on the same page

2.8.2 Panel Functions

Panel functions can be used to add additional features to a Trellis graph. For example, given a Trellis x-y plot, one can add a line using the panel function `panel.abline()`. For a list of available panel functions in R, type `?panel.functions` at the R prompt. For details with S-PLUS, see the S-PLUS Programmer's Guide.

Example 2.29 Create a Trellis x-y plot of `cm` versus `kg` given `Doctor` using the data frame `EPIDURAL`. Use panel functions to superimpose the ordinary least squares line and a least-trimmed squares line over the x-y plot.

Solution: The commands that follow create Figure 2.26.

```
> library(lattice)
> library(MASS)                                # Needed for lqs
> attach(EPIDURAL)
> xyplot(cm~kg|Doctor, as.table=TRUE,
+ panel=function(x, y)
+ {
+ panel.xyplot(x, y)                           # x-y plot
+ panel.abline(lm(y~x))                        # Least sq line
+ panel.abline(lqs(y~x), col=3, lty=2, lwd=2)  # Least trim sq line
+ }
+ )
```

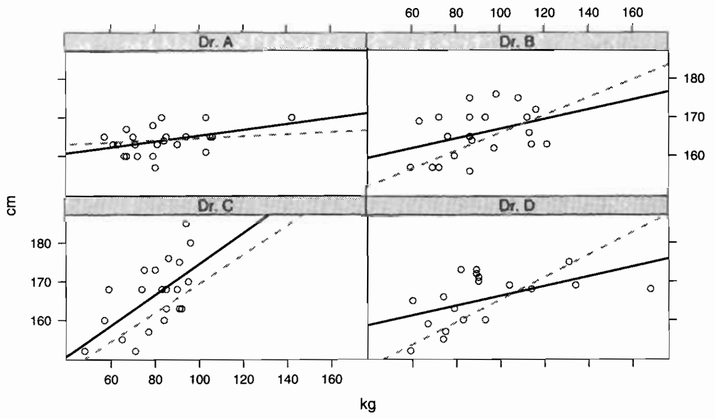

FIGURE 2.26: x-y plot of height (`cm`) versus weight (`kg`) given physician (`Doctor`) with superimposed least squares and least-trimmed squares lines

Another approach is to create a panel function that will superimpose the least squares and least-trimmed squares lines on an x-y plot and then to call that function within the `xyplot()` as follows:

```
> panel.scatreg <- function(x, y)              # name function
+ {
+ panel.xyplot(x, y)                           # make x-y plot
+ panel.abline(lm(y~x), lwd=2)                 # regression line
+ panel.abline(lqs(y~x), col=3, lty=2, lwd=2)  # least trim sq line
+ }
> xyplot(cm~kg|Doctor, as.table=TRUE, panel=panel.scatreg)
> detach(EPIDURAL)
```

Both approaches produce identical output. The dashed lines (`lty=2`) in Figure 2.26 are the least-trimmed squares lines. ■

2.9 Problems

1. Load the `MASS` package.

   (a) Enter the command `help(package="MASS")` and read about the functions and data contained in this package.

   (b) What does the description in the help file say about the function `lqs()`?
   Enter `help(lqs, package="MASS")` to obtain information about the command `lqs`.

   (c) What command shows the loaded packages?

2. Load `Cars93` from the `MASS` package.

   (a) Create density histograms for the variables `Min.Price`, `Max.Price`, `Weight`, and `Length` variables using a different color for each histogram.

   (b) Superimpose estimated density curves over the histograms.

   (c) Load the `lattice` package and do a box and whiskers plot of `Price` for every type of vehicle according to the drive train. Do you observe any differences between prices?

3. Load the data frame `WheatSpain` from the `PASWR` package.

   (a) Find the quantiles, deciles, mean, maximum, minimum, interquartile range, variance, and standard deviation of the variable `hectares`. Comment on the results. What was Spain's 2004 total harvested wheat area in hectares?

   (b) Create a function that calculates the quantiles, the mean, the variance, the standard deviation, the total, and the range of any variable.

   (c) Which communities are below the 10th percentile in `hectares`? Which communities are above the 90th percentile? In which percentile is Navarra?

   (d) Create and display in the same graphics device a frequency histogram of the variable `acres` and a density histogram of the variable `acres`. Superimpose a density curve over the second histogram.

   (e) Explain why using breaks of 0; 100,000; 250,000; 360,000; and 1,550,000 automatically results in a density histogram.

   (f) Create and display in the same graphics device a barplot of `acres` and a density histogram of `acres` using break points of 0; 100,000; 250,000; 360,000; and 1,550,000.

   (g) Add vertical lines to the density histogram of `acres` to indicate the locations of the mean and the median, respectively.

   (h) Create a boxplot of `hectares` and label the communities that appear as outliers in the boxplot. (Hint: Use `identify()`.)

   (i) Determine the community with the largest harvested wheat surface area using either acres or hectares. Remove this community from the data frame and compute the mean, median, and standard deviation of `hectares`. How do these values compare to the values for these statistics computed in (a)?

4. Load the `wheatUSA2004` data frame from the `PASWR` package.

(a) Find the quantiles, deciles, mean, maximum, minimum, interquantile range, variance, and standard deviation for the variable `ACRES`. Comment on what the most appropriate measures of center and spread would be for this variable. What is the USA's 2004 total harvested wheat surface are?

(b) Which states are below the 20th percentile? Which states are above the 80th percentile? In which quantile is WI (Wisconsin)?

(c) Create a frequency and a density histogram in the same graphics device using square plotting regions of the values in `ACRES`.

(d) Add vertical lines to the density histogram from (c) to indicate the location of the mean and the median.

(e) Create a boxplot of the `ACRES` and locate the outliers' communities and their values.

(f) Determine the state with the largest harvested wheat surface in acres. Remove this state from the data frame and compute the mean, median, and standard deviation of `ACRES`. How do these values compare to the values for these statistics computed in (a)?

5. The data frame `vit2005` in the `PASWR` package contains descriptive information and the appraised total price (in euros) for apartments in Vitoria, Spain.

(a) Create a frequency table, a piechart, and a barplot showing the number of apartments grouped by the variable `out`. For you, which method conveys the information best?

(b) Characterize the distribution of the variable `totalprice`.

(c) Characterize the relationship between `totalprice` and `area`.

(d) Create a Trellis plot of `totalprice` versus `area` conditioning on `toilets`. Are there any outliers? Ignoring any outliers, between what two values of `area` do apartments have both one and two bathrooms?

(e) Use the `area` values reported in (d) to create a subset of apartments that have both one and two bathrooms. By how much does an additional bathroom increase the appraised value of an apartment? Would you be willing to pay for an additional bathroom if you lived in Vitoria, Spain?

6. Access the data from url

http://www.stat.berkeley.edu/users/statlabs/data/babies.data

and store the information in an object named `BABIES` using the function `read.table()`. A description of the variables can be found at

http://www.stat.berkeley.edu/users/statlabs/labs.html.

These data are a subset from a much larger study dealing with child health and development.

(a) Create a "clean" data set that removes subjects if any observations on the subject are "unknown." Note that `bwt`, `gestation`, `parity`, `height`, `weight`, and `smoke` use values of 999, 999, 9, 99, 999, and 9, respectively, to denote "unknown." Store the modified data set in an object named `CLEAN`.

(b) Use the information in `CLEAN` to create a density histogram of the birth weights of babies whose mothers have never smoked (`smoke`=0) and another histogram placed directly below the first in the same graphics device for the birth weights of babies whose mothers currently smoke (`smoke`=1). Make the range of the x-axis 30 to 180 (ounces) for both histograms. Superimpose a density curve over each histogram.

(c) Based on the histograms in (b), characterize the distribution of baby birth weight for both non-smoking and smoking mothers.

(d) What is the mean weight difference between babies of smokers and non-smokers? Can you think of any reasons not to use the mean as a measure of center to compare birth weights in this problem?

(e) Create side-by-side boxplots to compare the birth weights of babies whose mother's never smoked and those who currently smoke. Use traditional graphics (`boxplot()`) as well as Trellis/lattice graphs to create the boxplots (`bwplot()`).

(f) What is the median weight difference between babies who are firstborn and those who are not?

(g) Create a single graph of the densities for pre-pregnancy `weight` for mothers who have never smoked and for mothers who currently smoke. Make sure both densities appear on the same graphics device and place a color coded legend in the top right corner of the graph.

(h) Characterize the pre-pregnancy distribution of `weight` for mothers who have never smoked and for mothers who currently smoke.

(i) What is the mean pre-pregnancy weight difference between mothers who do not smoke and those who do? Can you think of any reasons not to use the mean as a measure of center to compare pre-pregnancy weights in this problem?

(j) Compute the body weight index (BWI) for each mother in `CLEAN`. Recall that BWI is defined as kg/m^2 (0.0254 m= 1 in., and 0.45359 kg= 1 lb.). Add the variables weight in kg, height in m, and BWI to `CLEAN` and store the result in `CLEANP`.

(k) Characterize the distribution of BWI.

(l) Group pregnant mothers according to their BWI quartile. Find the mean and standard deviation for baby birth weights in each quartile for mothers who have never smoked and those who currently smoke. Find the median and IQR for baby birth weights in each quartile for mothers who have never smoked and those who currently smoke. Based on your answers, would you characterize birth weight in each group as relatively symmetric or skewed? Create histograms and densities of `bwt` conditioned on BWI quartiles and whether the mother smokes to verify your previous assertions about the shape.

(m) Create side-by-side boxplots of `bwt` based on whether the mother smokes conditioned on BWI quartiles. Does this graph verify your findings in (l)?

(n) Does it appear that BWI is related to the birth weight of a baby? Create a scatterplot of birth weight (`bwt`) versus BWI while conditioning on BWI quartiles and whether the mother smokes to help answer the question.

(o) Replace baby birth weight (`bwt`) with gestation length (`gestation`) and answer questions (l), (m), and (n).

(p) Create a scatterplot of `bwt` versus `gestation` conditioned on BWI quartiles and whether the mother smokes. Fit straight lines to the data using `lm()`, `lqs()`, and `rlm()`; and display the lines in the scatterplots. What do you find interesting about the resulting graphs?

(q) Create a table of `smoke` by `parity`. Display the numerical results in a graph. What percent of mothers did not smoke during the pregnancy of their first child?

7. Some claim the final hours aboard the RMS Titanic were marked by class warfare; others claim it was characterized by male chivalry. The data frame `titanic3` from the `PASWR`

package contains information pertaining to class status (`pclass`), survival of passengers (`survived`), and gender (`sex`), to name but a few. Based on the information in `titanic3`:

(a) Determine the fraction of survivors (`survived`) according to class (`pclass`).

(b) Compute the fraction of survivors according to class and gender. Did men in first class or women in third class have a higher survival rate?

(c) How would you characterize the distribution of `age`?

(d) Were the median and mean ages for females who survived higher or lower than for females who did not survive? Report the median and mean ages as well as an appropriate measure of spread for each statistic.

(e) Were the median and mean ages for males who survived higher or lower than for males who did not survive? Report the median and mean ages as well as an appropriate measure of spread for each statistic.

(f) What was the age of the youngest female in first class who survived?

(g) Do the data suggest class warfare, male chivalry, or some combination of both characterized the final hours aboard the Titanic? Feel free to explore other relationships based on the numbers in `titanic3` in answering this question.

8. Use the `Cars2004EU` data frame from the `PASWR` package which contains the numbers of cars per 1000 inhabitants (`cars`), the total number of known mortal accidents (`deaths`), and the country population/1000 (`population`) for the 25 member countries of the European Union for the year 2004.

(a) Compute the total number of cars per 1000 inhabitants in each country, and store the result in an object named `total.cars`. Determine the total number of known automobile fatalities in 2004 divided by the total number of cars for each country and store the result in an object named `death.rate`.

(b) Create a barplot showing the automobile death rate for each of the European Union member countries. Make the bars increase in magnitude so that the countries with the smallest automobile death rates appear first.

(c) Which country has the lowest automobile death rate? Which country has the highest automobile death rate?

(d) Create a scatterplot of `population` versus `total.cars`. How would you characterize the relationship?

(e) Find the least squares estimates for regressing `population` on `total.cars`. Superimpose the least squares line on the scatterplot from (d). What population does the least squares model predict for a country with a `total.cars` value of 19224.630? Find the difference between the population predicted from the least squares model and the actual population for the country with a `total.cars` value of 19224.630.

(f) Create a scatterplot of `total.cars` versus `death.rate`. How would you characterize the relationship between the two variables?

(g) Compute Spearman's rank correlation coefficient of `total.cars` and `death.rate`. (Hint: Use `cor(x, y, method="spearman")`.) What is this coefficient measuring?

(h) Plot the logarithm of `total.cars` versus the logarithm of `death.rate`. How would you characterize the relationship?

(i) What are the least squares estimates for the regression of `log(death.rate)` on `log(total.cars)`. Superimpose the least squares line on the scatterplot from (h). What death rate does the least squares model predict for a country with a `log(total.cars)` value of 9.863948? Make sure you express your answer in the same units as those used for `death.rate`.

9. The data frame `SurfaceSpain` in the `PASWR` package contains the surface area (km^2) for seventeen autonomous Spanish communities.

   (a) Use the function `merge()` to combine the data frames `WheatSpain` (from problem 3) and `SurfaceSpain` into a new data frame named `DataSpain`.

   (b) Create a variable named `surface.h` containing the surface area of each autonomous community in hectares. (Note: 100 hectares = 1 km^2.) Create a variable named `wheat.p` containing the percent surface area in each autonomous community dedicated to growing wheat. Add the newly created variables to the data frame `DataSpain` and store the result as a data frame with the name `DataSpain.m`.

   (c) Assign the names of the autonomous communities as row names for `DataSpain.m` and remove the variable `community` from the data frame.

   (d) Create a barplot showing the percent surface area dedicated to growing wheat for each of the seventeen Spanish autonomous communities. Arrange the communities by decreasing percentages.

   (e) Display the percent surface area dedicated to growing wheat for each of the seventeen Spanish autonomous communities using the function `dotchart()`. To read about `dotchart()`, type `?dotchart` at the command prompt. Do you prefer the barchart or the dotchart? Explain your answer.

   (f) Describe the relationship between the surface area in an autonomous community dedicated to growing wheat (`hectares`) and the total surface area of the autonomous community (`surface.h`).

   (g) Describe the relationship between the surface area in an autonomous community dedicated to growing wheat (`hectares`) and the percent of surface area dedicated to growing wheat out of the communities' total surface area (`wheat.p`).

   (h) Develop a model to predict the surface area in an autonomous community dedicated to growing wheat (`hectares`) based on the total surface area of the autonomous community (`surface.h`).