

Chapter 2

Data and Statistics

Statistics is the field of study whose objective is the transformation of data (usually sets of numbers along with identifying characteristics) into information (usually in the form of tables, graphs, and written and verbal summaries) that can inform sound policy decisions. We give examples of applications of statistics to many fields in Chapter 1. Here we focus on the general concepts describing the collection and arrangement of the numbers themselves.

2.1 Types of Data

Traditionally, we refer to five different *types* of data: count, categorical, ordered, interval, and ratio.

count data: The observational unit either has, or does not have, a particular property. For example, tossed coins can come up heads or tails. We count the number n of heads when a total of N coins are tossed.

categorical data: The data values are distinct from each other. Categorical variables are also referred to as *nominal* variables, *class* variables, or *factors*. The various categories or classes of a categorical variable are called its *levels*. An example of a factor, from the introductory paragraph of Chapter 6, is *factory* having six levels. That is, the investigation takes place at six factories. If we *code* *factory* as $\{1, 2, 3, 4, 5, 6\}$, meaning that we arbitrarily assign these six numbers to the six factories, we must be careful not to interpret these codes as ratio data. Coding the *factory* levels as integers doesn't give us the right to do arithmetic on the code numbers.

ordered data: The data values can be placed in a rank ordering. For any two observations, the analyst knows which of the two is larger, but not necessarily the magnitude of the difference between them. There is a distinct concept of *first*, *second*, ..., *last*. There is no way to measure the distance between values. An example is military ranks: A general is higher-ranked than a colonel, which in turn is higher than a major. There is no easy way to say something like, “A general is twice as far above a colonel as a colonel is above a major.”

interval data: The data values have well-defined distances between them, but there is not a ratio relationship. School grades are an example. Students in 10th grade have studied one year longer than students in 9th grade; similarly, students in 9th grade have studied one year longer than students in 8th grade. It is not meaningful to say a 10th-grade student is twice as knowledgeable as a 5th-grade student.

ratio data: The data values are measured by real numbers: There are a well-defined origin and a well-defined unit. Height of people is an example. There is a well-defined 0 height. We can speak of one person being 1 inch taller than another or of being 10% taller than another.

We also have another categorization of data as *discrete* or *continuous*. Discrete data have a finite or countably infinite number of possible values the data can take. Continuous data take any real number value in an interval; the interval may be either closed or open.

Many of the datasets we will study, both in this book and in the data analysis situations that this book prepares you for, have several variables. Frequently, there are one or more ratio-scaled numeric variables associated with each value of a categorical variable. When only one numeric variable is measured for each observational unit, the dataset is said to be *univariate*. When there are k ($k > 1$) variables measured on each observational unit, the dataset is said to be *multivariate*. Multivariate datasets require additional techniques to identify and respond to correlations among the observed variables.

2.2 Data Display and Calculation

Data are often collected and presented as tables of numbers. Analysis reports are also frequently presented as numbers. Tables of numbers can be presented on a page in ways that make them easier to read or harder to read. We illustrate some of each here and will identify some of the formatting decisions that affect the legibility of numerical tables.

2.2.1 Presentation

There are two general principles:

alignment of decimal points: Units digits of each number are in the same vertical column. Larger numbers extend farther to the left.

display comparable numbers with common precision: Numbers to be compared are displayed so the positions to be compared are in the same column.

Table 2.1 shows two tables with identical numerical information. The first is legible because it follows both principles; the second is not because it doesn't.

Table 2.1 Legible and illegible tabular displays of the same numerical data: In panel a the numbers are aligned on the decimal point and are displayed to the same precision (the same number of decimal digits). In panel b the numbers are centered or left justified—with the effect of hiding the comparability, and displayed with different precisions—which further hides comparability.

a. Legible			b. Illegible		
109.209	133.502	112.219	109.209	133.50234	112.21
153.917	78.971	109.311	153.9	78	109.31152
80.269	83.762	77.036	80.26	83.76253	77.036
74.813	112.720	119.719	74.81323	112.72001	119.7
84.228	103.849	85.586	84.2	103.	85.58
80.558	100.944	115.134	80.55801	100.94474	115.13436
85.519	89.280	109.247	85.51940	89.28095	109.24788

2.2.2 Rounding

The number of decimal digits in a number indicates the precision with which the number was observed. Any analysis normally retains the same precision. Any changes in the number of decimal digits that are not justified by the method of analysis implicitly suggests that the data are either more or less precise than they actually are. This can lead to misleading inferences and wrong policy decisions.

Please see Appendix G for an illustration of the potential problems and references to more detailed discussion. Be sure to read FAQ 7.31 in file

```
system.file(".././doc/FAQ")
```

The help menus in **Rgui** in Windows and **R.app** on Macintosh have direct links to the FAQ file.

There are simple rules:

1. DO NOT ROUND intermediate values! Keep all 16 digits of double precision arithmetic in a computer program and all 12 digits on pocket calculators. For example, if a correct calculation $7.1449/3.6451 = 1.9601$ is rounded to $7.14/3.65$, the quotient is less than 1.96 and a decision based on whether or not the result exceeds 1.96 will reach an incorrect conclusion.
2. Final answers may be rounded to the SAME number of significant digits as the original data. You may never have final answers with more digits than any intermediate value or with more digits than the original data.
3. Standard deviations can give a hint as to the number of believable digits. For example, if $\bar{x} = 1.23456$ and $s_{\bar{x}} = .0789$, then we can justifiably round to $\bar{x} \approx 1.234$ (using the logic that $t = 1.23456/.0789 = 15.64715 \approx 15.647$ is good to three decimal positions).

2.3 Importing Data

R, and other statistical software systems, have functions that can read data in a variety of formats. All the datasets used in this book are included in the **HH** package. Section 2.3.1 tells how to access them.

Access to datasets in other formats and in other locations (anywhere on the internet) is described in Section 2.3.2.

2.3.1 Datasets for This Book

We have many datasets that we analyze in examples or make available for analysis in exercises. Most datasets are real, taken from journal articles; data repositories of governments, corporations and organizations; data libraries; or our own consulting experience. Citations to these datasets are included in the text. As befits a text, most data we present are structured for the techniques of the chapter in which we present it. Our datasets are frequently used in more than one chapter. We have an Index of Datasets with which you can locate all references to a specific dataset across chapters.

The datasets discussed in this book are available for readers in two different formats.

For use with R, all datasets mentioned in the book are available in the **HH** package for R. The **HH** package can be downloaded from CRAN (Comprehensive R

Archive Network) for use on any computer on which **R** is installed. Details on installing **R** are in Appendix A. Once the **HH** is loaded into an **R** session, the ABCD dataset is made accessible with the statement

```
data(ABCD)
```

Additional information on the **HH** package is in Appendix B.

For use with any other software system, the datasets mentioned in the book are available in ASCII format as *csv* files. These are text files in which each row of data appears on one row of the file. Within a row, the items are separated by commas. Further discussion of the ASCII files, including the *url* where they are available, is in Appendix H.

2.3.2 Other Data sources

In consulting environments data is often collected and stored in a database management system. **R** has packages that can read directly from database management systems that may be housed anywhere on the internet.

Datasets are often stored in MS Excel *xls* files. These can be directly read into **R** on any operating system using the **XLConnect** package. See Section A.1.4 for further discussion. On MS Windows machines, the **RExcel** software is available for direct interaction between **R** and Excel. See Appendix D for further information.

Datasets stored as datafiles in the internal format of other statistical software systems may be migrated to an **R** analysis. **R** can read and write most of them with the aid of the **foreign** package. See the help file `help(package="foreign")` for further information.

2.4 Analysis with Missing Values

Statisticians frequently encounter situations where an analysis cannot be completed in a straightforward fashion because some portion of the data is missing. For some analyses, missing or unbalanced data cause no difficulty beyond the need to calculate results with a revised formula. Examples include the two-sample *t*-test of Section 5.4.3 and the one-way analysis of variance of Chapter 6. In other circumstances, such as multiple regression analyses discussed in Chapters 9 to 11, the analyst must either discard the observations carrying incomplete information or use sophisticated techniques beyond the scope of this book to impute, or estimate, the missing portions of the existing data. If the reasons for “missingness” are related to the problem being addressed, abandoning observations is likely to lead to incorrect inferences. If the data are missing at random, discarding a few observations may be a satisfactory solution, but the smaller the ultimate sample size, the less likely the analysis will

produce useful and correct results. Imputing the values of missing data is usually preferable to discarding cases with incomplete information. We recommend Little and Rubin (2002) as a comprehensive reference on how to handle missing data, particularly techniques of imputation.

A discussion of how missing values are handled in R is in Section 2.A.

2.5 Data Rearrangement

Datasets are not necessarily arranged in the most convenient way for the analysis you have in mind. Rearrangement is usually easy. Frequently the functions in the **reshape2** package will be helpful.

We usually work with one of the two data arrangements. Table 2.2 shows both arrangements and the use of the **reshape2** functions `melt` and `dcast` to convert between them. One arrangement (the `data.frame` wide in Table 2.2) is a set of multiple columns (`x` and `y`), one per variable, with factor levels `Names` explicitly indicated by data values in the appropriate column. Each observation has all its values listed in the same row of all columns.

The other (the `data.frame` long in Table 2.2) contains all the numeric values in a single column (`value`), with levels of factors explicitly identified in their own columns (`Names` and `variable`). Note that the two different variables in the wide arrangement are represented by two levels of the `variable` factor in the long arrangement.

2.6 Tables and Graphs

Graphs constructed from data arranged in a table are generally more useful and informative than the table. The human eye and brain can quickly discern patterns from a well-constructed picture that would be far from obvious from the underlying tabular data. Excellent examples are contained in Tufte (2001) and Wainer (1997).

Characteristics that we wish to reveal with our graphs are location, variability, scale, shape, correlation, interaction, clustering, and outliers. In Chapter 4 we illustrate many of these characteristics, primarily through our discussion of scatterplots and scatterplot matrices. Additional types of displays are presented in many subsequent chapters. We discuss both the information about the data that we obtain from the graphs and the structure of the graphs. We introduce many new types of graphs throughout the book. In the appendix to Chapter 4 we provide a summary on those new graphs that are based on Cartesian products.

2.7 R Code Files for *Statistical Analysis and Data Display* (HH)

The **HH** package is available for R from CRAN. See Appendix A for details on installing R with our recommended packages on your computer. The **HH** package includes all datasets used in the book. R scripts for all figures and tables in the book are included in files in the **HH** package. See Appendix B for details. Many of the graphs were produced with functions that are included and fully documented in the **HH** package.

Table 2.2 Define `wide`, a `data.frame` in the wide arrangement. Convert it to the long arrangement with the `melt` function, and convert it back with the `dcast` function.

```
> library(reshape2)

> wide <- data.frame(Names=LETTERS[1:5], x=1:5, y=6:10)

> wide
  Names x  y
1     A 1  6
2     B 2  7
3     C 3  8
4     D 4  9
5     E 5 10

> long <- melt(wide, id="Names")

> long
  Names variable value
1     A         x     1
2     B         x     2
3     C         x     3
4     D         x     4
5     E         x     5
6     A         y     6
7     B         y     7
8     C         y     8
9     D         y     9
10    E         y    10

> wideagain <- dcast(Names ~ variable, value="value", data=long)

> wideagain
  Names x  y
1     A 1  6
2     B 2  7
3     C 3  8
4     D 4  9
5     E 5 10
```

The R code for all *examples*, and for occasional *exercises* is included with the **HH** package from CRAN. Thus you can duplicate all tables and figures in the book and you can use these as templates for analyzing other datasets. The R code for the examples in each chapter of the Second Edition is in a file named after the chapter. For example, the code file for Chapter 6, the one-way analysis of variance chapter, is in file `hh2/oway.R`. The full path to the file on your computer is found by entering

```
HHscriptnames(6)
```

at the R prompt. The content of the tables and figures is not available as files. They may all be reproduced by running the code.

The R code for the First Edition is also available. Enter

```
HHscriptnames(6, edition=1)
```

at the R prompt.

The Second edition code is identical to the code that actually produced the tables and figures. The book was written using the `Sweave` and `Stangle` functions from the **utils** package with the L^AT_EX document preparation system. See Appendix N for links to L^AT_EX. All code is included within the L^AT_EX source for the book. See `help(Sweave, package="utils")` for details on writing using Sweave.

For the reader of this book, all you need to know is how to find the code for a chapter (`HHscriptnames(6)` as indicated above), and the structure of the files. Each file starts with a line that tells the name of my L^AT_EX source file for that chapter. It then has code *chunks*, with each chunk being the code associated with a table or figure. The first chunk in all files is the line

```
library(HH)
```

Each file is independent of all other chapters and assumes only that the **HH** package is loaded. Multiple chunks associated with the same dataset in the same file assume the previous chunks have already been run.

The chunks begin with function calls to the `hhpdf` or `hhcapture` functions. When I was writing the book, these calls were defined to capture the figure or table as a file. For the reader, these calls are defined in the **HH** package as *noops* (NO OPERATION)—that is, they don't do anything. All output goes directly to your console window or your graphics window.

The best way to use the files is to pick up their lines and paste them in the R console window. It will often be helpful to study how the lines are constructed.

It is possible to `source` the entire file. While it works, all it does is produce all the tables and figures that you already have in the book. Sourcing the files won't help in learning.

2.A Appendix: Missing Values in R

The R convention for missing values is NA (a standard abbreviation for “Not Available” or “No Answer”). When R knows that a value is missing it prints “NA” (without the quotes). When R is reading an ASCII data file, it will recognize by default the character sequence “NA” as a missing observation.

If the ASCII data file uses some other convention (such as the “.” that SAS uses by default), then we must tell R to use a different convention for reading missing values either with an argument to the `read.table` function or, after the reading, by some logical investigation of the data values.

R has several conventions for working with datasets containing NA values.

Data Input: See Tables 2.3, 2.4, and 2.5 for an example. We use the default missing value indicator in Table 2.3, an explicitly defined missing value indicator in Table 2.4, and a non-default missing value indicator in Table 2.5 without telling the `read.table` function that we were doing so.

Table 2.3 The data are read from a `text` argument, which is equivalent to reading from a text file. In the AA example, the default missing value is NA. In the BB example in Table 2.4, the argument `na.strings` defines strings “999” and “.” to indicate missing values. The internal representation is the R value NA. In the CC example in Table 2.5, where we didn’t use the argument `na.strings`, the `y` variable has been coerced to be a factor. Read the help file

```
help("read.table", package="utils")
```

for more information

```
> AA <- read.table(text="
+ x y
+ 1 2
+ 3 NA
+ 5 6
+ ", header=TRUE)

> AA
  x y
1 1 2
2 3 NA
3 5 6

> sapply(AA, class)
      x      y
"integer" "integer"
```

Table 2.4 The argument `na.strings` defines strings "999" and "." to indicate missing values.

```

> BB <- read.table(text="
+ x y
+ 1 2
+ 3 999
+ 5 6
+ 7 .
+ 9 10
+ ", header=TRUE, na.strings=c("999", "."))

> BB
  x y
1 1 2
2 3 NA
3 5 6
4 7 NA
5 9 10

> sapply(BB, class)
      x      y
"integer" "integer"

```

Table 2.5 We neglected to use the argument `na.strings`. The `y` variable has become a factor.

```

> CC <- read.table(text="
+ x y
+ 1 2
+ 3 999
+ 5 6
+ 7 .
+ 9 10
+ ", header=TRUE)

> CC
  x y
1 1 2
2 3 999
3 5 6
4 7 .
5 9 10

> sapply(CC, class)
      x      y
"integer" "factor"

> CC$y
[1] 2 999 6 . 10
Levels: . 10 2 6 999

```

Printing: Missing numerical values are displayed as NA. Missing character and factor items are displayed as <NA>. See [Table 2.6](#)

Table 2.6 Missing numerical values are displayed as NA. Missing character and factor items are displayed as <NA>.

```
> abcd <- data.frame(x=c(1, 2, NA, 4, 5, 6, 7, 8),
+                   y=c(6, 5, 8, NA, 10, 9, 12, 11),
+                   ch=c(NA, "N", "O", "P", "Q", "R", "S", "T"),
+                   stringsAsFactors=FALSE)
```

```
> abcd
```

	x	y	ch
1	1	6	<NA>
2	2	5	N
3	NA	8	O
4	4	NA	P
5	5	10	Q
6	6	9	R
7	7	12	S
8	8	11	T

```
> sapply(abcd, class)
```

	x	y	ch
	"numeric"	"numeric"	"character"

Graphs: Points whose coordinates are not known (points “O” and “P”) are not printed. Points with known coordinates and unknown value (the first point whose value should have been “M”) are displayed as NA in the known position. See Figure 2.1.

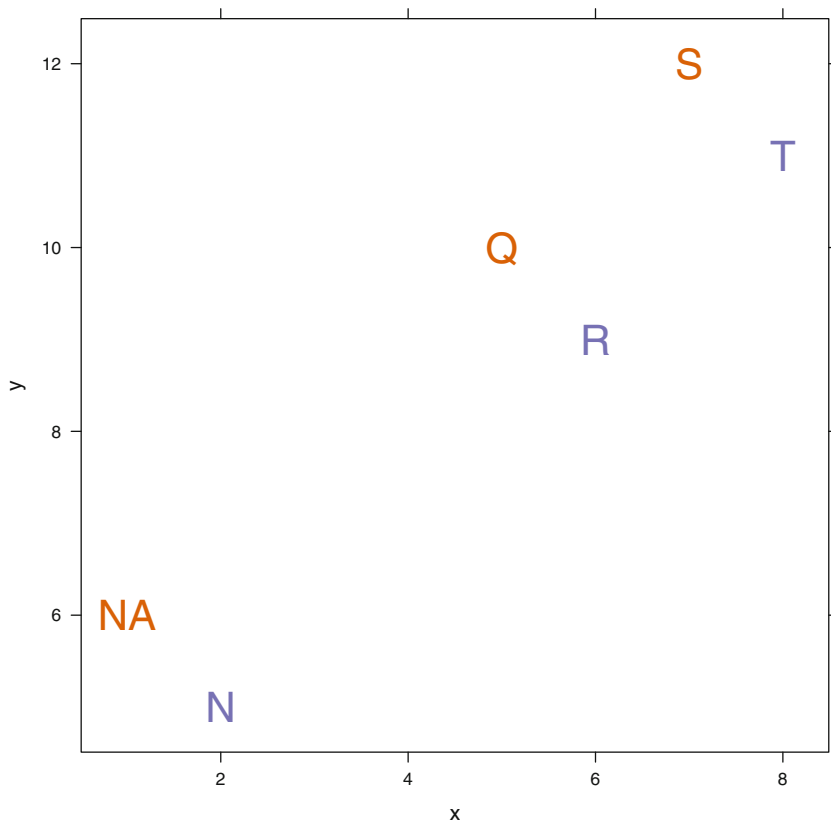


Fig. 2.1 The dataset `abcd` is defined in Table 2.6. The plot was drawn with

```
> xyplot(y ~ x, data=abcd, labels=abcd$ch, panel=panel.text,
+        col=c("red", "blue"))
```

Arithmetic: Arithmetic with missing values returns a missing value. Many functions, `sum` is illustrated in Table 2.7, can be told to remove the missing values and sum the non-missing values.

Table 2.7 Arithmetic with missing values returns a missing value. Many functions, `sum` and `mean` are illustrated in Table 2.7, can be told to remove the missing values and sum the non-missing values.

```
> 3 + NA
[1] NA

> sum(3, NA)
[1] NA

> sum(3, NA, na.rm=TRUE)
[1] 3

> abcd$x
[1] 1 2 NA 4 5 6 7 8

> mean(abcd$x)
[1] NA

> mean(abcd$x, na.rm=TRUE)
[1] 4.714
```

Linear Models: Default (`na.action=na.omit`) behavior is to remove the row. Table 2.8 shows the default behavior of the `lm` and related modeling functions. The entire row containing the missing values is removed from the analysis and subsequent processing. Table 2.9 shows an optional better behavior.

Table 2.8 The default behavior of the `lm` and related modeling functions. The entire row containing the missing values is removed from the analysis and subsequent processing. See Table 2.9 for an optional better behavior.

```
> a.lm <- lm(y ~ x, data=abcd)

> summary(a.lm)

Call:
lm.default(formula = y ~ x, data = abcd)

Residuals:
    1     2     5     6     7     8 
0.704 -1.219  1.013 -0.910  1.167 -0.755 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.373      1.053    4.16  0.0142 *
x              0.923      0.193    4.79  0.0087 **
---
Signif. codes:
  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.2 on 4 degrees of freedom
(2 observations deleted due to missingness)
Multiple R-squared:  0.851, Adjusted R-squared:  0.814 
F-statistic: 22.9 on 1 and 4 DF, p-value: 0.00872

> predict(a.lm)
    1     2     5     6     7     8 
5.296  6.219  8.987  9.910 10.833 11.755
```

Linear Models: Better behavior (`na.action=na.exclude`) is to keep track of which rows have been omitted. Table 2.9 shows an optional better behavior. The rows with missing values are still removed from the calculations of the "lm" object, but information on which rows were suppressed is retained.

Table 2.9 With `na.action=na.exclude`, the rows with missing values are still removed from the calculations of the "lm" object, but information on which rows were suppressed is retained.

```
> b.lm <- lm(y ~ x, data=abcd, na.action=na.exclude)

> summary(b.lm)

Call:
lm.default(formula = y ~ x, data = abcd, na.action = na.exclude)

Residuals:
    1      2      5      6      7      8 
0.704 -1.219  1.013 -0.910  1.167 -0.755 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.373      1.053     4.16  0.0142 *
x               0.923      0.193     4.79  0.0087 **
---
Signif. codes:
  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.2 on 4 degrees of freedom
(2 observations deleted due to missingness)
Multiple R-squared:  0.851, Adjusted R-squared:  0.814 
F-statistic: 22.9 on 1 and 4 DF, p-value: 0.00872

> predict(b.lm)
    1      2      3      4      5      6      7      8 
5.296  6.219   NA   NA  8.987  9.910 10.833 11.755
```

<http://www.springer.com/978-1-4939-2121-8>

Statistical Analysis and Data Display

An Intermediate Course with Examples in R

Heiberger, R.M.; Holland, B.

2015, XXXI, 898 p. 341 illus., 326 illus. in color.,

Hardcover

ISBN: 978-1-4939-2121-8