Appendix A
Some Basics of R

A.1 Installing R

The first order of business is to download and install R on your computer (if you haven't already). Make sure you are connected to the Internet, and go to the R web site, http://www.r-project.org/. This site is worth exploring. You will find considerable useful information here concerning R including a list of books treating elementary and advanced topics in statistics with extensive examples in R. But right now our goal is to download and install the most recent version of R that is appropriate for your type of computer.

To access the site for downloading, click on the link labeled `CRAN` on the list at the left of the page. `CRAN` stands for Comprehensive R Archive Network. Then, choose a mirror site that is geographically close to you.

R may be compiled from source but it is strongly recommended to begin by installing a pre-compiled, binary version. Choose your platform by clicking on one of Linux, Mac OS X or Windows and follow any links to the binaries for the base distribution. For example, in the case of Linux, you will need to proceed further to choose your distribution. Download the binary and follow normal procedures for installation. In the case of Windows and Mac OS X, this means just double clicking on the icon for the binary installation and following the instructions. Linux users will follow the installation procedures that are specific to their distribution, for example using `rpm` for Fedora distributions. After a few minutes, depending on the speed of your Internet connection, you will be ready to open an R session.

Running R will depend on your platform. For Windows and Mac OS X, there is a GUI interface that can be opened by simply double clicking on the R icon that has been installed by the above procedures. Under unix (e.g., linux and in a terminal window under Mac OS X), R can be run from a terminal shell command line by entering the command "R." A popular option under unix is to run R from within an editor that has macro and syntax coloring capabilities, such as emacs using the ESS (Emacs Speaks Statistics) macros, providing a more powerful interface than the command line on its own. Documentation can be found by searching for "ESS"

K. Knoblauch and L.T. Maloney, *Modeling Psychophysical Data in R*, Use R! 32,
DOI 10.1007/978-1-4614-4475-6,

on http://www.r-project.org/ under "Related Projects" or more directly at the ESS homepage http://ess.r-project.org/. `Rstudio` (http://rstudio.org/), a relatively new project at the time of this writing, provides a platform independent GUI with a uniform interface and a number of additional features to help the user interact with R.

You only need to download and install an executable copy of R by following the instructions above. This copy includes extensive documentation, about 30 packages that are ready to be loaded and used in an R session, including a large collection of sample data sets used in illustrating commands.

A.2 Basic Data Types

Everything in R is an object, even the functions that operate on them. Here, we introduce some of the basic object types in R and how to manipulate them. A more detailed and definitive explanation will be found in the documentation that comes with R.

A.2.1 Vectors

The simplest data objects in R are vectors. These are indexed sets of elements of all the same class. The simplest vector types are called atomic and the most frequently encountered of these have types "numeric," "integer," "character," "logical." There are also atomic vectors of type "complex" and "raw."

Vectors are created in numerous ways. The simplest uses the function `c`

```
> ( x <- c(1.2, 4, 5.1, 7.2, pi) )

[1] 1.20 4.00 5.10 7.20 3.14
```

which here assigns a 5-element vector of class "numeric" to the variable `x`. Note that the variable `pi` is predefined in R. There are no scalar variables, but only vectors of length 1.

```
> ( y <- 5 )
[1] 5
```

In a similar fashion, we can create vectors of other classes.

```
> a <- c("Over", "the", "rainbow")
> d <- x < 4
> ls.str()
a :   chr [1:3] "Over" "the" "rainbow"
d :   logi [1:5] TRUE FALSE FALSE FALSE TRUE
x :   num [1:5] 1.2 4 5.1 7.2 3.14
y :   num 5
```

We created two vectors, respectively, of class "character" and "logical" that we will use in subsequent examples in this section. We, also, used the function `ls.str` to show the structures of the four vectors that we have added to the workspace. The vector `a` has three elements, character strings, each of class "character."

Note how we also created a logical vector `d` from a conditional operation on a vector of another class. In mathematical operations, the logical values of `FALSE` and `TRUE` take on the numeric values of 0 and 1, respectively.

```
> sum(d)
[1] 2
```

R contains standard comparison operators for creating logical vectors, such as >, <, >=, etc. We note that the negation operator in R is `!` and the "not equal" operator `!=`. To test equality, we use the double equal sign, ==, as in many other computer languages. However, equality comparisons between "numeric" variables should be performed with circumspection because of digital limitations in the representation of floating point numbers.

```
> sqrt(3)^2 == 3
[1] FALSE
```

For this, R has the `all.equal` function to compare objects of near equality.

```
> all.equal(sqrt(3)^2, 3)
[1] TRUE
```

R contains two operators for logical AND. The first is `&` which performs comparisons element-by-element between a pair of vectors of numeric, logical or complex type. When used on numeric or complex values, a nonzero entry counts as `TRUE`, a zero entry as `FALSE`.

```
> c(1,0,2,0) & c(1,2,3,4)
[1]  TRUE FALSE  TRUE FALSE
```

The operator `&&` is used for a comparison between a single pair of elements numeric, logical, or complex variables. If `&&` is applied to a pair of vectors, only the first pair of elements from each will be compared.

```
> c(1,0,2,0) && c(1,2,3,4)
[1] TRUE
```

which would probably not be the intended purpose. A similar pair of operators exists for OR operations: `|` for pairwise comparison of vector elements and `||` for a single comparison.

There are numerous other functions that are useful for generating vectors with specific structure. To generate sequences,

```
> seq(1, 10, 2)
[1] 1 3 5 7 9
```

```
> seq(0, 1, length.out = 5)
[1] 0.00 0.25 0.50 0.75 1.00
```

or sequences that include repeated values

```
> rep(a, 3)
[1] "Over"    "the"     "rainbow" "Over"    "the"
[6] "rainbow" "Over"    "the"     "rainbow"
> rep(a, each = 3)
[1] "Over"   "Over"    "Over"    "the"     "the"
[6] "the"    "rainbow" "rainbow" "rainbow"
> rep(a, 1:3)
[1] "Over"   "the"    "the"   "rainbow" "rainbow"
[6] "rainbow"
```

using the character vector `a` previously defined on p. 305. The `:` operator is a shortcut for `seq(1, 3)` for which the third argument takes the default value of 1. The various forms of the second argument of `rep` allow one to generate different patterns of repetition. We will introduce and use these and similar functions throughout the book for coding explanatory variables that will be useful in simulation and modeling.

All vector types can include missing data indicated by the value `NA` (for not available).

```
> ( z <- c(3.1, 1.8, NA, 0.24) )
[1] 3.10 1.80   NA 0.24
```

Thus, the elements of logical vectors can be in any of three states {`TRUE`, `FALSE`, `NA`}. Some care must be taken with missing data, however, as all operations with input including an `NA` yield `NA`. Many functions include an argument `na.rm` that, when set to `TRUE`, conveniently causes the missing values to be ignored.

```
> mean(z)
[1] NA
> mean(z, na.rm = TRUE)
[1] 1.71
```

The elements of vectors are accessed using square brackets, as in

```
> x[4]
[1] 7.2
> a[c(2, 3)]
[1] "the"     "rainbow"
> x[ x > 4]
[1] 5.1 7.2
```

In the second and third examples above, the indices are a vector, permitting selection of several elements at once. In the third example, this vector was generated by a condition, which results in the selection of the elements for which the condition is TRUE. A particularly useful feature is the use of negative indices to specify which elements to exclude.

```
> v <- 1:10
> v[-seq(1, 5, 2)]
[1]  2  4  6  7  8  9 10
```

Positive and negative indices cannot be mixed, however.

There are several special functions for accessing the structure of the elements of a character string. For example,

```
> nchar(a[3])
[1] 7
> substring(a[1], 2, 4)
[1] "ver"
```

the effects of which should be obvious from the examples.

Ordinary arithmetic operations work element by element on vectors of the same length.

```
> x + 1:5
[1]  2.20  6.00  8.10 11.20  8.14
> x * rep(c(2, 3), c(3, 2))
[1]  2.40  8.00 10.20 21.60  9.42
```

With vectors of different lengths, the shorter one is automatically recycled, but a warning will be generated if the shorter length is not a multiple of the longer one.

```
> x + y
[1]  6.20  9.00 10.10 12.20  8.14
> x + 1:2
[1] 2.20 6.00 6.10 9.20 4.14

Warning message: In x + 1:2 :
 longer object length is not a multiple of shorter object length
```

A.2.1.1 Attributes and Indexing

A powerful feature of R is that objects may have *attributes* attached to them. These can influence how the object is treated by functions. For example, a vector can have a names attribute that allows the elements to be indexed by these names.

```
> w <- c(a = 5, b = 16, c = 4.1)
> w
```

```
   a    b    c
 5.0 16.0  4.1
> w["b"]
 b
16
> w[letters[2:3]]
   b    c
16.0  4.1
```

The names attribute affects how vectors are printed (i.e., by how they are displayed by their print method, implicitly called when we give the variable name). We, also, see the use of a predefined vector, `letters`, that contains the set of lowercase letters, to index the named elements of `w`. A similar vector `LETTERS` is predefined for the uppercase letters.

In R, matrices are just vectors that have a `dim` attribute.

```
> u <- 1:8
> dim(u) <- c(2, 4)
> u
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

The `dim` attribute has even changed the class of the vector, which you can check with the function `class`. The matrix is filled column by column.

Arrays are vectors with more than 2 dimensions.

```
> dim(u) <- rep(2, 3)
> class(u)
[1] "array"
> str(u)
 int [1:2, 1:2, 1:2] 1 2 3 4 5 6 7 8
```

Although assigning a `dim` attribute suffices for transforming a vector to a matrix (array), normally one would create a matrix (array) using the `matrix` (`array`) function(s).

```
> m <- matrix(1:9, nrow = 3, ncol = 3)
> arr <- array(1:16, dim = c(2, 2, 4))
```

Matrix elements are indexed or subset as with vectors using brackets but with an index for each dimension.

```
> m[2, 3]
[1] 8
> m[3, ]
[1] 3 6 9
```

```
> m[, -2]
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
```

Leaving a dimension blank is a shortcut for including all of its values, and negative indexing can be used as well.

Matrix multiplication is defined by the operator `%*%` as in

```
> m %*% m[, -1]
     [,1] [,2]
[1,]   66  102
[2,]   81  126
[3,]   96  150
```

but, of course, only between matrices with conforming dimensions.

A.2.2 Factors

Factors are a special class of vector that are used for representing categorical variables. The factor can take on values from a finite set of categories called levels. The levels are attached to the variable as an attribute, but it is not necessary that all of the levels occur in the vector. For example, suppose that we wanted to categorize observers with respect to their type of color vision. The levels might be

```
> ColVisLevs <- c("Normal", "Protan", "Deutan", "Tritan")
```

and we define a factor

```
> ColVisType <- factor(rep(ColVisLevs, c(5, 1, 2, 0)),
+      levels = ColVisLevs)
> ColVisType
[1] Normal Normal Normal Normal Normal Protan Deutan Deutan
Levels: Normal Protan Deutan Tritan
```

We use the function `factor` to coerce a character vector to class "factor." The level "Tritan" does not occur in our vector but using the `levels = ColVisLevs` argument, we ensure that our factor has four levels.

Ordinarily, factors are printed in terms of the labels given to their levels, but, in fact, underlying the labels, the levels are coded as integers, here 1–4. This is revealed with the `unclass` function.

```
> unclass(ColVisType)
[1] 1 1 1 1 1 2 3 3
attr(,"levels")
[1] "Normal" "Protan"  "Deutan" "Tritan"
```

The argument `levels = ColVisLevs`, specified in the `factor` function above, ensured that the levels were assigned to the codes in the order given, rather than the default alphabetical order. The difference between the underlying codes of the factors and the labels of their levels is critical to keep in mind when manipulating factors, or error will surely ensue. The great value of factors is in modeling data. They are used in specifying categorical variables in a specialized language for specifying models in a simplified formula (e.g., see Chap. 2) and exploited internally by modeling functions to construct the design matrix for estimating the model parameters.

In some cases, categorical variables can be ordered. For example, a finer categorization of color deficient observers might indicate them as "simple anomalous," "extreme anomalous," and "dichromatic." There is a natural order among these categories in terms of capacity to make fine discriminations in certain parts of color space, which parts depending upon which type of color deficiency. Such information is included in a factor using the function `ordered`.

```
> DiscriminationLevs <- c("simple", "extreme", "dichromat")
> ColDiscType <- ordered(DiscriminationLevs[c(1,
+     1, 3, 2, 1, 2, 2, 3)], levels = DiscriminationLevs)
> ColDiscType
[1] simple    simple    dichromat extreme   simple
[6] extreme extreme   dichromat
Levels: simple < extreme < dichromat
```

Examining the class of such objects indicates that they have both classes "ordered" and "factor," so that they inherit the properties of ordinary factors. They are of value primarily when modeling. For example, we will use them in Chap. 3 for the representation of observer ratings.

A.2.3 Lists

Vectors, matrices, and arrays are limited to containing elements of all the same type. Lists, however, may contain objects of different types, even other lists.

```
> a.lst <- list(A = LETTERS[1:5], B = (1:5)^2,
+   state = c(TRUE, FALSE, TRUE),
+   f = factor(c("Male", "Female", "Male", "Female", "Female")))
> a.lst
$A [1] "A" "B" "C" "D" "E"

$B [1]  1  4  9 16 25

$state
[1]  TRUE FALSE  TRUE
```

```
$f [1] Male   Female Male   Female Female
Levels: Female Male
```

`a.lst` contains four components, each one a vector of a different class and length. There are several ways to access the components. Here, the components are named, but if they are not, then individual components can be accessed using double brackets.

```
> a.lst[[2]]
[1]  1  4  9 16 25
```

which in this case returns a vector of "numeric." Unlike for vectors and matrices, only one index, not a range, is permissible. Some care is required because single brackets may be used, but they return a list whose first component is the object, not the object itself.

```
> a.lst[2]
$B [1]  1  4  9 16 25
```

which returns a list with component `B`, a vector of "numeric."

When the components are named, they may be accessed using the `$` operator.

```
> a.lst$state
[1]  TRUE FALSE  TRUE
```

The name may also be used with the double brackets if quoted.

```
> a.lst[["state"]]
[1]  TRUE FALSE  TRUE
```

Lists are quite flexible and many of the exotic objects one runs into in R are simply lists that have been assigned a specific class and for which specific method functions have been written.

A.2.4 Data Frames

The data frame is a class that is ubiquitous in R because it conveniently structures the data in diverse circumstances. In essence, it is a list with components that are vectors of all the same length. A data frame is usually created with the `data.frame` function. Here is a simple example,

```
> d.df <- data.frame(x = c(0.12, 0.15, 0.11, 0.13, 0.10, 0.25,
+          0.22, 0.27),  Sex = c("Female",  "Male",  "Female",
+          "Female", "Male", "Male", "Male",  "Male"),
+          CVT = ColVisType
+          )
> d.df
```

```
     x    Sex    CVT
1 0.12 Female Normal
2 0.15   Male Normal
3 0.11 Female Normal
4 0.13 Female Normal
5 0.10   Male Normal
6 0.25   Male Protan
7 0.22   Male Deutan
8 0.27   Male Deutan
```

This data frame has three components, a "numeric" vector `x`, and two factor vectors, `Sex` and `CVT` that we already defined above. Note that we did not specify that `Sex` would be a factor but it was coerced to one automatically. Because the components all have the same length, the data frame has a rectangular structure. It is displayed by its print method in tabular format. Typically, each row will correspond to an experimental observation and each column to a characteristic of the observation, such as the response and values of the observation on other explanatory variables, covariates, and factors.

Because it is a list, a data frame can be accessed by any of the methods used for lists.

```
> d.df$Sex
[1] Female Male Female Female Male Male Male Male Levels:
Female Male
> d.df[[1]]
[1] 0.12 0.15 0.11 0.13 0.10 0.25 0.22 0.27
> d.df[["CVT"]]
[1] Normal Normal Normal Normal Normal Protan Deutan Deutan
Levels: Normal Protan Deutan Tritan
```

Additionally, because of their tabular structure, methods have been defined so that data frames can also be accessed as if they were matrices.

```
> d.df[5, 3]
[1] Normal Levels: Normal Protan Deutan Tritan
> d.df[, 1]
[1] 0.12 0.15 0.11 0.13 0.10 0.25 0.22 0.27
> d.df[4, ]
     x    Sex    CVT
4 0.13 Female Normal
```

Note, however, that the first two examples above return vectors whereas the third returns a 1-row data frame. This makes sense since the items in a column are all of the same type, as each column *is* a vector. In contrast, different columns are likely to be of different types, preventing a row from being represented as a vector.

A data frame can be created manually as above or read-in from data files, as we will see in Sect. A.5.1.

A.2.5 Other Types of Objects

The descriptions above cover the built-in data types that one is most likely to encounter when beginning to use R. A few others are treated in the main text (e.g., formula objects and functions). Other more exotic types of built-in objects, such as environments, will not be treated in this book.[1] It can be useful to understand something about them, however, and the reader is encouraged to read the documentation that comes with R to learn more as necessary. There is a much larger set of derived objects defined as the output to modeling functions. A complication is that there are two object systems operational in R, designated as S3 and S4.

Many of the objects defined with respect to the S3 system are simply lists that have been assigned a special class. Specific methods for such a class will typically be defined to allow manipulating and extracting information from such objects, although in the event that a method does not exist, components can be accessed as if the object were an ordinary list.

The S4 system is more formalized. Objects defined using S4 methods have *slots* rather than components. Slots are accessed using the `@` symbol rather than the `$` symbol. Knowledge of S4 methods beyond this will not be necessary in this book.

A.3 Simple Programming Constructs

For programming, R includes standard conditional constructs, such as `if` and `else` for testing a single condition, but also `ifelse` for testing a vector.

```
> x <- 1:6
> ifelse(x > 3, "red", "green")
[1] "green" "green" "green" "red"   "red"   "red"
```

The `switch` function is used to choose an action based on multiple values of a variable.

Iterative looping can be programmed with `for`, `while` and `repeat` constructs. R also includes a family of `apply` functions, `apply`, `tapply`, `sapply`, etc., that can loop over matrices, factors, lists, etc. Examples of their use and explanation are scattered throughout the book.

A.4 Interacting with the Operating System

It is generally recommended to consecrate a separate directory for each project that you undertake in R. This permits keeping data sets and scripts related to the same project together. The function `getwd` returns the path to the current directory as

[1] The work space is an environment, as are each of the elements on the search path.

a character string, and the function `setwd` will change the working directory to the path name given by a character string as its first argument. The file names in the working directory are returned in a character vector by the function `dir` but it can also be used to return the names or the names with their path from the current directory or names containing a particular pattern of text. This is particularly useful in automating the read-in and analysis of sets of data files all at once.

Another useful function for interacting with the operating system is `system` which takes a system command as a text string and runs it in the operating system. It may return the result to R if so specified. See its help page. Also, the function `pipe` can be used to read data to and from a connection given as a character string (See Sect. A.5.7).

A.5 Getting Data into and Out of R

Nearly all the data sets in this book are retrieved either from packages, such as the package for this book, **MPDiR**, or generated by simulation. Most packages contain sample data sets for demonstration of the types of data to which their methods apply. In addition, R comes by default with over 100 data sets available in memory (try `ls("package:datasets")`) that provide a great variety of sample data with which to play, with which to model and from which to learn. Many packages use the `LazyData` option, so that the data sets are available by name on the search path as data frames as soon as the package has been loaded with the `library` function. If the package does not exploit the `LazyData` option, then data sets can still be read into the workspace using the `data` function, as demonstrated throughout the book. Alternatively, data sets can be read into the workspace without first loading the package by specifying the package name as an additional argument. For example, to load the `HSP` data set into the workspace without loading the **MPDiR** package, one executes

```
> data(HSP, package = "MPDiR")
```

where the package name is given as a character vector.

Most users will want to work on their own data, however, and will require understanding how to store on disk and retrieve them into memory. R is equipped with a diverse set of functions for accessing different formats and sources of data. In practice, just a few of these functions will probably serve nearly all of your needs. In the rest of this chapter, we will describe some of the facilities in R for accessing data. For a more thorough introduction, see the document "R Data Import/Export" which comes with any R installation. Nearly all books on R cover this material to some extent, but in particular see [132, 162] for a comprehensive exposition of the subject.

A.5.1 Writing and Reading Text Files

A.5.2 write.table

Text files are extremely versatile for storing data as they can be created and accessed by many different programs. They provide a simple medium for storing a data frame and restoring it to the work space. To write a data frame to a storage medium, use the `write.table` function. For example, the following code suffices to store `HSP` on disk, in a text file, supposing that it is still in the work space.

```
> write.table(HSP, file = "HSP.txt")
```

The name of the object and the file name to which it will be written must be specified. If no file by that name exists in the working directory, then one will be created. If one exists, it will be overwritten, unless the argument `append = TRUE` is included. This argument allows the user to append multiple data frames to one file. There are several other arguments that allow additional control of the format of the output file, for example, specifying the field separator, the end of line character, whether row and column names should be included, etc. As usual, see the help page for the details.

The output file is an ordinary text file that can be edited or displayed. If the argument `row.names = FALSE` is specified, then the file structure is rectangular with the same number of fields on each line, possibly with the column names as the first line. If row and column names are included in the output, then the first line of column names will include one less field (it has no row name). This presents no problem as the `read.table` command to which we will now turn will automatically interpret the first line as column names when it has one fewer field than the succeeding lines.

A.5.3 read.table

As indicated above, to read in a text file in a tabular format, possibly with a header line, the `read.table` function is extremely versatile. To read in the file that we just saved is as simple as

```
> HSP <- read.table("HSP.txt")
> str(HSP)
'data.frame':   30 obs. of  5 variables:
 $ Q  : num  46.9 73.1 113.8 177.4 276.1 ...
 $ p  : num  0 9.4 33.3 73.5 100 100 0 7.5 40 80 ...
 $ N  : int  35 35 35 35 35 35 40 40 40 40 ...
 $ Obs: Factor w/ 3 levels "MHP","SH","SS": 2 2 2 2 2 2 2 ..
 $ Run: Factor w/ 2 levels "R1","R2": 1 1 1 1 1 1 2 2 2 2 ..
```

The first argument is the file name specified as a character string. The succeeding function call shows that `read.table` returns a data frame. The character variables are read in as factors by default, but character variables may, in appropriate circumstances, be read-in as logical, complex, or even numeric. If "character" is the preferred class for the character columns in the file, then the argument `as.is = TRUE` should be included. The function distinguishes between columns that contain numbers with a decimal point, read in as numeric, and those without, read in as integers. If the text file does not have a column of row numbers, and the first line is a row of column names, then the argument `header = TRUE` should be included. The columns will be given default names `V1, V2, ...` when `header = FALSE` is specified. The help page for `read.table` describes 22 possible arguments that provide a great deal of flexibility in the format of files that it may read, including options that permit handling non-rectangular file formats.

The first argument is not limited to file names of text files, however. R deals with input and output devices through objects with class "connection." The first argument can refer to any connection that can be read-in as text. The function will automatically open the connection, read in the data, and close the connection. For example, the following code uses a URL to read a data set based on the Stiles and Burch color matching fundamentals from the Colour & Vision Database (http://www.cvrl.org/) over the web.

```
> SB2deg.df <- read.table(
+   "http://www.cvrl.org/database/data/cones/linss10e_5.csv",
+   sep = ",")
```

Note that the extension of the file indicates that the fields are comma separated so we include the `sep = ","` argument; alternatively, we could have used the `read.csv` function directly.

Beginning with line 47, the fourth column of the data set is blank, reflecting the dichromacy of human color matches in this portion of the spectrum. The data for these values are read in as `NA`. We can substitute zeros for the NA values by either

```
> SB2deg.df[is.na(SB2deg.df[, 4]), 4] <- 0
```

or since these are the only `NA` values in the data frame,

```
> SB2deg.df[!complete.cases(SB2deg.df), 4] <- 0
```

where the function `complete.cases` conveniently returns a logical vector indicating the rows that do not contain values of `NA`. We, therefore, negate this vector with the negation operator, "`!`," to select the rows that do.

Another very useful operation with `read.table` is to read data from the clipboard, the buffer in which data are temporarily stored during a copy operation from the keyboard. Under Windows the user can type

```
> read.table("clipboard")
```

and the contents of the last copy operation will be returned. An alternate syntax is also available under Mac OS X when not using X11.

```
> read.table(pipe("pbpaste"))
```

`pbpaste` is a command, only available under Mac OS X, that writes the clipboard to stdin. The function `pipe` returns the contents of stdin. A complementary function `pbwrite` will permit writing to the clipboard as well. These operations are described under `help(connections)`. Reading from the clipboard is probably the simplest way to transfer data from a spreadsheet program to R or from a browser window. However, in the latter case, one must beware of hidden character codes that can mess-up the transfer.

A.5.4 `scan`

Sometimes one wants to access data that are simpler than a table, for example, a sequence of numbers or characters in a file or a matrix. The function `scan` provides a simple means of accomplishing this task. It takes a file name or text connection as first argument. By default, it assumes that the input is numeric, but this can be set otherwise by the `what` argument. If successful, it returns a vector of the given type. If reading a matrix from a file, the input can be formatted directly in R but care must be taken because matrices are filled column-wise, by default. `scan` can be used to read from the clipboard using the same arguments as described above.

A.5.5 `save` *and* `load`

Table formatted files do not suffice to store all types of data used by R. Objects can have attributes, which are arbitrary lists attached to them. These are not saved in the typical text representation. Also, arbitrary lists and objects cannot be written and read with the above described functions.

The function `save` writes an object or a series of objects to a file, by default in a compressed binary format. The stored representation is an exact copy of the object(s), with, for example, attributes intact. Typically, such files are given an extension of ".Rdata" and are transferrable across different computing platforms. The data files used throughout this book were created using `save` but with the extension ".rda."

The data in a file created with `save` can be retrieved with the function `load`, which minimally takes the filename as its argument. On some operating systems (e.g., Mac OS X), a double mouse click on the file icon that had been saved in this fashion is sufficient to open the R GUI (if not already open and) to run the `load` function.

The functions `dput` and `dump` can be used to create ASCII representations of objects more exotic than a data frame. However, it is not guaranteed that certain characteristics of the object will be restored when sourced into memory.

The Posting Guide (at http://www.r-project.org/posting-guide.html) outlines suggested etiquette for interacting with the online R-Help mailing lists and suggests using `dump` to communicate complex data structures to the list.

A.5.6 Interacting with Matlab

One of the major platforms for analyzing data from experiments in vision (and possibly in other psychophysical domains) is Matlab and, thus, a great quantity of data is likely stored in ".mat" files.

The package **R.matlab** [12] provides `readMat` and `writeMat` functions for files of this format. As of Matlab v7, the data compression scheme changed and, in order to read these files, one will also need the **Rcompression** [106] package available from the Omegahat repository (http://www.omegahat.org/R).[2]

The **R.matlab** package also includes facilities for setting up a client/server relationship between R and Matlab. One can then transfer data back and forth between R and Matlab and R may send commands to Matlab, as well. This functionality is only available for Matlab v6 or higher.

A.5.7 Interacting with Excel Files

We indicated in Sect. A.5.3 how to transfer data from a spreadsheet to R through a keyboard copy operation. Another strategy is to save the file from Excel in a "comma separated value" format (with a ".csv" extension). Then, it can be read in with one of the functions `read.csv` or `read.csv2` which are simply wrapper functions for `read.table` with the appropriate default arguments. The second version of the function is set up to read formats in which a comma is used as the decimal point.

Several packages cater to reading and writing directly from Excel spreadsheet files. The data must generally be in tabular format, however. The package **gdata** [181] includes a `read.xls` function, and the packages **WriteXLS** [160] and **dataframes2xls** [163] write files in a format that can be opened by Excel. The package **XLConnect** allows more sophisticated interactions between R and Excel [70].

A.5.8 Reading in Other Formats

Data files come in many formats, and it sometimes seems as if every statistical package has its own special data encryption, thereby complicating transfer of

[2] As of version 1.5.0 of this package, **RCompression** is only necessary for R versions older than 2.10.0.

data between platforms. The recommended package **foreign** [145] contains several functions for reading from and writing to formats for several popular statistical packages.

Databases with their associated query languages provide a powerful means of storing, accessing, and manipulating data in a manner that can be independent of the statistical package operating on the data. They are of some interest in R as providing a buffer for dealing with data sets that would exhaust or strain computer memory. There is a growing number of packages that cater to reading and writing to databases. The list, **DBI** [81], **RJDBC** [177], **RMySQL** [83], **RODBC** [149], **ROracle** [84], **RPostgreSQL** [39], **RSQLite** [82], is a sample obtained by a search on the term "database" at the CRAN package web site (http://cran.r-project.org/web/packages/).

Appendix B
Statistical Background

B.1 Introduction

In this appendix we summarize without proof some basic results of mathematical statistics that we use in the main chapters of the book. We do assume that the reader has some previous experience with elementary probability theory and statistics and is familiar with hypothesis testing and linear regression, etc. Any undergraduate statistics book would cover enough material to serve as the prerequisite.

There is no need to read this appendix from start to finish. There are numerous references to sections of this appendix in each chapter and the reader interested in a particular chapter can read it, referring to material in this appendix only as needed.

The models used in simulating psychophysical observers are typically parametric, i.e., they are completely specified except for a small number of "free" parameters. The experimenter who summarizes his (or her) data set by estimating the mean and variance of a data sample is effectively fitting a normal (Gaussian) distribution and can use the fitted distribution to estimate confidence intervals, quantiles, etc. Of course, there are other distributions besides the Gaussian, and we will review some of them below.

Any possible model of the observer results from selecting values of the free parameters and the first task in any psychophysical application is to select the parameter values so that the model best matches a subject's behavior in a particular experimental condition.

Figure B.1a, for example, is a histogram of 75 choice response times of a single observer from a perceptual task. On each trial, the observer viewed an array of dots in which pairs of the dots could be oriented either randomly or along virtual concentric circles, forming a Glass Pattern, in the latter case [69]. The observer's task was to decide as rapidly as possible whether the concentric pattern was present or not. Each response is the time in seconds between the start of a trial and the touch response.

The Gaussian distribution would be a poor choice as a model for these response times since they appear to be skewed (asymmetric), and we know from long

K. Knoblauch and L.T. Maloney, *Modeling Psychophysical Data in R*, Use R! 32,
DOI 10.1007/978-1-4614-4475-6,

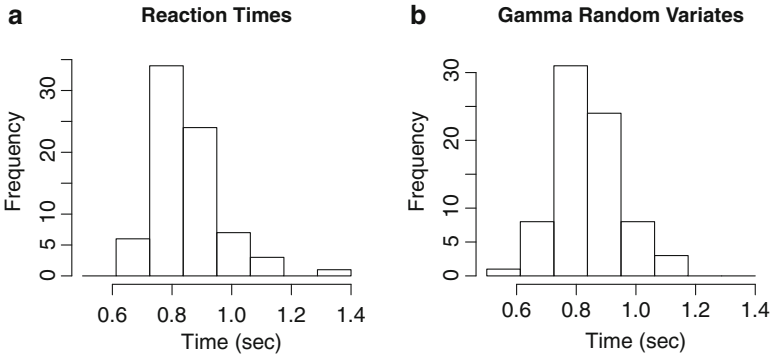

Fig. B.1 (**a**) Response times in a choice response task. Unpublished data provided by Elodie Mahler. (**b**) Histogram of values simulated from a Gamma distribution with best fit parameters to the data in part (**a**)

experience that response times are often skewed. We first select a parametric model of the observer's choice reaction times and in this case we use the family of Gamma distributions with two free parameters. In Fig. B.1b we show the results for a simulated observer based on the Gamma distribution in R, a parametric family of distributions. We selected the parameters so that the histogram of simulated data, generated by numerical methods with R (using the `rgamma` function; see below), resembles the actual histogram. The number of data points for the simulated and actual observers is the same. If we have succeeded in matching the model to the observer, then the simulated data "could have been" the actual data.

Of course, the model of the observer on which we base a simulation is, first of all, an hypothesis concerning the observer, that a Gamma distribution is an appropriate model, and, second of all, a claim that we have estimated the free parameters of the Gamma that "best" match the observer. Stating the model clearly and comparing it to actual data allows us to test and, if need be, revise the model. Perhaps there is no Gamma distribution that adequately captures human performance. We may also have a number of possible models of the observer in mind, and we are trying to decide which model offers the best description of the observer's behavior, a process referred to as *model selection* [24, 206]. This appendix summarizes basic results from mathematical statistics concerning *parametric model fitting* and *parametric model selection*.

The fitting problem is the heart of statistics. What is novel here is the emphasis we place on building actual simulations of observers based on explicit models of the observer. Of course, some such model is typically presupposed in any parametric estimation or model selection problem. Even if we chose to analyze data using a standard statistical package and a standard method such as multiple regression, there are models of how data were generated underlying and justifying the fitting method.

There are several advantages to emphasizing simulation in the way that we do. If, for example, the experimenter cannot specify such a simulation program, then his

failure may mean that he has not thought through his assumptions about what the observer is doing in the task. Or, even though he can write the simulation program, he may find that he has serious misgivings about the model as representative of human performance, once it is written down and all of its assumptions are made clear. With many statistical packages, for example, it is easy to fit data to very sophisticated models without paying much attention to assumptions about the model implicit in the fitting procedure. It is currently too easy to fit data to indefensible models.

Last, if we accept that the simulation is a surrogate for the actual experiment, the experimenter can test out different experimental designs and fitting methods before collecting data. Like a good athlete, the experimenter can "practice his moves" before the actual competition. He may discover, for example, that his experimental design is not adequate to answer the questions that interest him and that an experimental design involving more trials is needed. These sorts of questions come under the heading of "power" in mathematical statistics [129] and the simulation-fitting loop is a convenient tool for analyzing the power of statistical tests. We will find that the simulation-fitting loop has other advantages. In particular it allows us to routinely use resampling methods and the bootstrap to estimate confidence intervals and biases for parameter estimates [55].

B.2 Random Variables and Monte Carlo Methods

Since Fechner [59], human responses in psychophysical tasks are often modeled by *random variables*. We are primarily concerned with two kinds, *finite discrete random variables* and *continuous random variables*. We concentrate on univariate random variables, where the random variable takes on only single real numbers (they are *univariate real*), although we also consider random variables that return a vector of real numbers. Most books on probability theory or statistics will define several types of random variables and their properties [34,153]. The handbooks by Johnson, Kotz, and colleagues [85–87] are encyclopedic treatments of commonly used (and not so commonly used) random variables and are very useful. We recommend them as bedtime reading. We will define only the minimum number of random variables needed in the remainder of a book.

B.2.1 Finite, Discrete Random Variables

A univariate finite discrete random variable X is defined by listing the set of values that it can take on $\{x_1, x_2, \ldots, x_n\}$ and the probability $p_i = p(x_i) \geq 0$ that each value can occur. The probabilities are constrained to sum to 1, $\sum_{i=1}^{n} p_i = 1$ ("something always happens"). The function $p(x_i)$ is referred to as the *probability mass function (pmf)* of the random variable.

B.2.2 Bernoulli Random Variables

For our purposes, the most important example of a finite discrete random variable is a *Bernoulli random variable B* with values $\{0,1\}$ and corresponding probabilities defined by $P[B=1]=p$ and $P[B=0]=1-p$. The *Bernoulli random variable* is a useful model for binary decisions between two possibilities (such as signal present, signal absent) where the outcome effectively is just a choice between two alternatives. The importance of the Bernoulli random variable is just that experimenters like to design experiments where the observer's response is limited to Yes/No, Present/Absent, Right/Left, Salty/Sweet, etc. We will further discuss the Bernoulli random variable and a generalization of it, the Binomial, below.

Simple as it is, the Bernoulli is an example of a *parameterized family* of random variables that we could denote by $\mathscr{B}(p)$ which includes Bernoulli random variables for all choices of the parameter $0 \leq p \leq 1$. If we specify p in $\mathscr{B}(p)$, then we have specified a particular random variable in the family. Later on, when we consider parametric estimation problems, we will try to decide which random variable in a family is the best match to an observer given his or her performance in a psychophysical task.

B.2.3 Continuous Random Variables

A univariate continuous random variable X is specified by giving its *probability density function (pdf)* $f(x) \geq 0$ with

$$\int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1. \tag{B.1}$$

The pdf effectively determines the probability that the random variable will be in any interval (a,b) with $a<b$ (see Fig. B.2)

$$P[X \in (a,b)] = P[a<X<b] = \int_a^b f(x)\,\mathrm{d}x. \tag{B.2}$$

Probability density should not be confused with probability. For a continuous random variable, the probability that it takes on a particular value a is 0, in symbols, $P[X=a]=0$. The evident paradox is that the random variable must take on *some* value, but each particular value has probability 0. These sorts of anomalies make probability theory confusing at first and the best approach to the subject involves suppressing intuitions and paying close attention to definitions. For us then, a pdf is a way to specify the probability that the random variable is in any interval and no more.

We will usually denote random variables by uppercase letters, for example, X, and the values that a random variable can take on by the corresponding lower-case

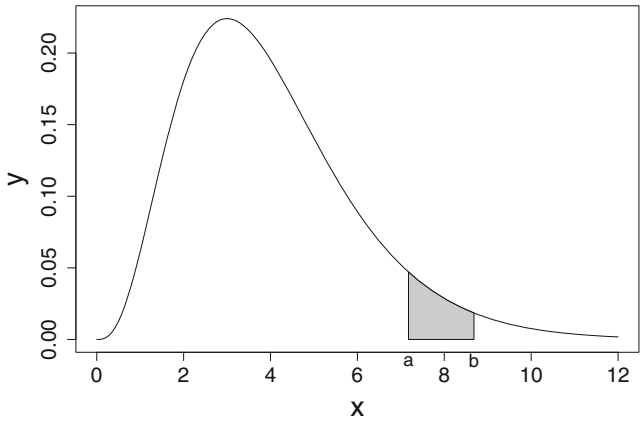

Fig. B.2 The probability that a random variable will be in the interval (a,b) is the area under the pdf over this interval

letter, e.g. x. The *cumulative distribution function* (cdf) of a continuous random variable is defined as

$$F_X(x) = \int_{-\infty}^{x} f(x)\mathrm{d}x \tag{B.3}$$

and since $f(x) = F'(x)$, either the pdf or cdf can be used to characterize a continuous random variable. The cdf of a discrete random variable is defined in the same way. The inverse of the cdf function of a random variable is referred to as the *quantile function*. If, for example, $F(x)$ is a distribution, then $F^{-1}(0.75)$ is the value of x that falls at the 0.75 quantile (0.75th percentile) of the distribution.

B.2.4 Discrete vs Continuous Random Variables

The distinction between discrete and continuous random variables is artificial. In more advanced probability texts the pmf of a discrete random variable would be written as a pdf in terms of Dirac (impulse) generalized functions $\delta(x)$ [19]. We can write the pdf of a finite discrete random variable with outcomes $x_1, \ldots, x_n$ and corresponding probabilities as

$$p(x) = \sum_{i=1}^{n} p_i \delta(x - x_i) \tag{B.4}$$

The resulting pdf takes on the value p_i precisely at x_i and is 0 otherwise. The single great advantage of using Dirac notation is that we can refer to finite discrete random

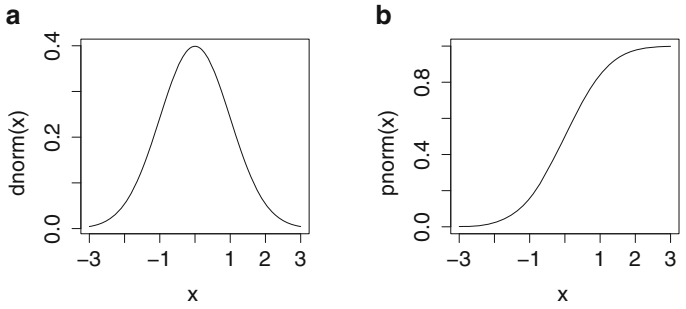

Fig. B.3 The pdf and cdf of a normal random variable with mean $\mu = 0$ and $\sigma = 1$

variables and continuous random variables and other types of random variables that are partly discrete and partly continuous using the same terminology and definitions.

We can define random variables (X, Y) that are bivariate and take values in the real plane. We define the pdf (pmf) by a function $f(x, y) \geq 0$ with

$$1 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \mathrm{d}x\, \mathrm{d}y \tag{B.5}$$

and we can define cdfs, inverse cdfs, etc. for such random variables. The extension to multivariate random variables is easily made but we will have little use for multivariate random variables $(X_1, \ldots, X_n)$ except to occasionally use multivariate notation and write the pdf as $f(x_1, \ldots, x_n)$.

B.2.5 Normal (Gaussian) Random Variables

If the reader has previously encountered a random family of continuous distributions, it is almost certainly the Gaussian. In the **stats** package, a recommended package that is normally attached to the search path at startup by default, four functions are defined for manipulating and simulating from each of a selection of 18 random variables. These four functions conventionally are prefixed with the letters "d," "p," "q" and "r," for the density (pdf), probability (cdf) and quantile functions (inverse cdf) and for random variate generation, respectively. All the functions for any random variable follow this same convention and all four functions are documented in the same help file. For example, if we type

```
> help("rnorm")
```

we obtain documentation for the four functions, `dnorm`, `pnorm`, `qnorm`, and `rnorm` for working with normal (Gaussian) random variables. The function `dnorm`, for example, is the pdf of the normal distribution while `pnorm` is the cdf. These are both plotted in Fig. B.3 using the default values of their arguments. `qnorm`

is the inverse cdf or quantile function $F^{-1}(x)$. If $p = F(x)$, then $x = F^{-1}(p)$. For example,

```
> qnorm(pnorm(1))
[1] 1
```

and

```
> pnorm(qnorm(0.8413447))
[1] 0.841
```

We can invert $F(x)$ only when it is a strictly increasing (or decreasing) function and the solution to the equation $p = F(x)$ is unique. However, if we are careful (and we always are) we can define an inverse for any cdf. We saw the use of the quantile function in estimating thresholds in Sect. 1.3.2.

B.2.6 Location-Scale Families

Recall that we are interested in parametric families of distributions. The normal family is a two-parameter family with parameters μ and $\sigma > 0$ that are referred to, respectively, as the "mean" and "standard deviation" of the normal random variable. In the call to `pnorm` above, $\mu = 0$ and $\sigma = 1$ are the default values. These defaults are overridden by adding additional arguments, for example, `pnorm(x, mean = 2, sd = 0.5)`. These parameters are examples of location and scale parameters, respectively, and the names are self-explanatory, at least as soon as we plot the pdf for several choices of $\mu = 0, 1, 2$ with $\sigma = 1$ and for $\sigma = 1, 2, 3$ and $\mu = 0$. The pdf is "located" at μ and its width is scaled by choice of σ.

We can take any pdf, $f(x)$, and turn it into a location-scale family by defining $f(x; \mu, \sigma) = f\left(\frac{x-\mu}{\sigma}\right)$. If $f(x)$ is already in a parameterized family, we just add μ and σ to the list of parameters. In the case of a Bernoulli random variable $B(p)$ that takes on values $\{0, 1\}$, we could change it to a generalized Bernoulli $B(p, \mu, \sigma)$ that takes on values $\{a, b\}$ with probabilities $1 - p, p$, respectively. For example, with $\mu = -0.5$ and $\sigma = 0.5$ the generalized Bernoulli now takes on the value 1 with probability p and otherwise -1. Notice that, as the scale parameter is $1/2$, the Bernoulli has been "stretched" so that the spacing between the two possible outcomes is 2, not 1.

In any location-scale family, the location parameter determines the location of the cdf as well. In Fig. B.4a we show the cdf of the normal for different values of μ and $\sigma = 1$. The scale parameter also scales the cdf. In the normal case (Fig. B.5b), different values of σ affect the steepness of the rise of the cdf near $\mu = 0$. A little bit of calculus reveals that the slope of the tangent to the cdf of the normal at μ is $(\sqrt{2\pi}\sigma)^{-1}$. The cdfs of location-scale random variables will be useful for modeling performance in psychophysical tasks, such as in estimating *psychometric functions*.

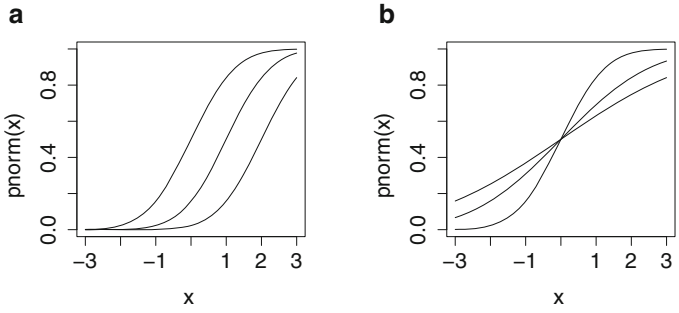

Fig. B.4 Normal distribution cdfs differing in (**a**) location and (**b**) scale

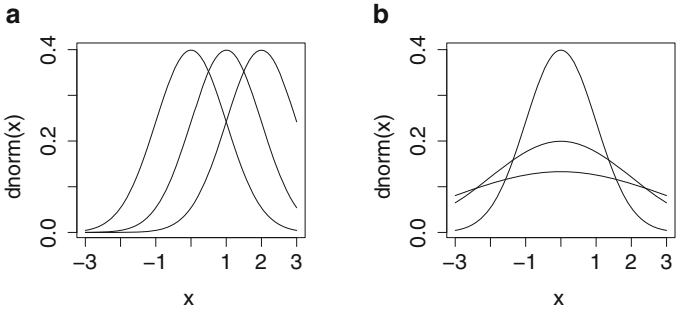

Fig. B.5 Normal distributions differing in (**a**) location and (**b**) scale

B.3 Independence and Samples

The lastfunction in the collection of R functions associated with the normal distribution is `rnorm`. It simulates a realization or "draw" of a normal random variable. We can think of a random variable as a box with a button on the side. Every time we push the button a real number pops out of the box.[1] We do not know in advance what number will pop out (that's why it is a *random* variable), but the pdf (or cdf) characterizes how likely it is that the realization is in any specified range. The first argument of `rnorm` "`n`" is the size of the *sample* desired, how many times we want to press the button. A sample is denoted by subscripted uppercase letters such as $X_1, \ldots, X_n$. This notation is ambiguous. It could refer to the future sample that we have not taken yet or the sample already collected which is a list of specific numbers. We will live with this ambiguity and we will call attention to it when it might cause confusion.

[1] Actually, only those real numbers that can be represented as floating point numbers in the computer.

A sample is, by definition, a series of *independent* draws from the same random variable. A definition of independence can be found in almost any probability book; intuitively, the elements in the sample are independent in that any one outcome is unaffected by the outcomes of the other draws. Alternatively, knowledge of some values of a sample tells us nothing about the remaining values that we did not know. We can also say that a sample is *independent, identically-distributed (iid)* to emphasize that all the sample values are drawn from the same distribution.

A sample can be considered as a multivariate random variable, $(X_1, \ldots, X_n)$, and independence is equivalent to the claim that its pdf is just $f_n(x_1, x_2, \ldots, x_n) = f(x_1)f(x_2), \ldots, f(x_n)$: that is, the multivariate pdf of the sample is just the product of n copies of the pdf $f(x)$ of the univariate random variable X. If X is part of a parameterized family with parameter θ then $f_n(x_1, x_2, \ldots, x_n; \theta) = f(x_1; \theta)f(x_2; \theta), \ldots, f(x_n; \theta)$.

As an example of a sample, we can type

```
> (X <- rnorm(10, mean = 0, sd = 1))
 [1]  0.0911  1.2501 -0.3081 -0.1473  0.7015  0.6408 -0.1959
 [8]  0.5594 -0.5278 -1.3072
```

and we obtain 10 values drawn from the normal distribution with parameters 0 and 1. A second draw,

```
> (Y <- rnorm(10, mean = 10, sd = 1))
 [1] 8.47 11.33  9.25  8.61  9.16  9.07  8.86  9.00 11.69  8.21
```

produces very different values, "roughly" 10 units greater than the first sample.

Independence is one of the deepest concepts in statistics and probability. Paradoxically, it is very unlikely that any sample of data from a psychophysical observer is really independent or identically distributed since the brain is, in effect, a large bowl of neural soup where everything affects everything and each trial permanently changes the brain and the outcome of the next. There are standard methods such as randomization of the order of trials to minimize the consequences of sequential effects. See Kingdom and Prins [91]. In a recent article, Fründ et al. [66] analyze the consequences of sequential dependence in psychophysical data.

B.3.1 Expectation and Moments

Given a random variable X with pdf $f(x)$, and any function $g(x)$, we compute the *expected value* of $g(X)$ as

$$\mathsf{E}[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)\,\mathrm{d}x \tag{B.6}$$

If the random variable is discrete, with outcomes x_i and probabilities p_i for $i = 1,\ldots,n$, the equation immediately above becomes

$$\mathsf{E}[g(X)] = \sum_{i=1}^{n} g(x_i)p_i. \tag{B.7}$$

If $g(x) = x$, then the resulting expected value is the expected value of X, sometimes referred to as the *population mean* and often denoted μ_X. If $g(x) = (x-\mu_X)^2$, then the resulting expected value is the *variance* of X. The expected value $\mathsf{E}[(X-\mu_X)^n]$ is the nth central moment of X. The variance of X is also its second central moment and is often denoted σ_X^2.

When a random variable is part of a parametric family, expectations may depend on the parameters of the family. For example, it is reassuring that, for a normal random variable X from the family $\mathcal{N}(\mu,\sigma^2)$, $\mathsf{E}(X) = \mu$ and $\mathrm{Var}(X) = \sigma^2$. We must be doing something right.

We can approximate the integral in (B.6) numerically [143], or we can approximate it by Monte Carlo methods as follows. If, for example, we wanted to know the fourth central moment of a normal random variable $\mathcal{N}(0,1)$, then we can simply take a very large sample using `rnorm(n, mu = 0, sigma = 1)`, compute $g(X) = (X-\mu)^4$, and average these values. A sample of size $n = 1{,}000{,}000$ gives us a Monte Carlo estimate of 3.004, not too far from the true value of 3. We will use Monte Carlo methods extensively to explore and approximate complex formulas involving random variables. After a while, we will think of calculus as the refuge of those with small computers.

In using computational methods in this way we are making use of a second key idea in probability theory, *convergence*. Large samples, in a very precise sense, characterize the distribution they are drawn from. We return to this point below in discussing *resampling methods*.

If the integral in (B.6) does not converge, then $E[g(X)]$ is not defined. In particular, not every random variable has a mean or variance. A good example of such a distribution is the Cauchy. The reader is invited to a estimate its nonexistent mean by Monte Carlo methods with progressively larger samples, using the function `rcauchy`.

We define expectation for bivariate and multivariate random variables similarly [34] and we can define bivariate and multivariate moments as well. For example, given the bivariate random variable (X,Y) with pdf $f(x,y)$, and any function $g(x,y)$, we can compute the *expected value* of $g(X,Y)$ as

$$\mathsf{E}[g(X,Y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)f(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{B.8}$$

If $g(X,Y) = X$, the resulting expected value is the *marginal expected value* of X denoted μ_X and μ_Y is defined similarly. If $g(X,Y) = (X-\mu_X)(Y-\mu_Y)$, the resulting expected value is the *covariance* of X and Y denoted $\mathrm{Cov}(X,Y)$. See, for example, [153] or any introductory statistics text for details.

We can define multivariate expectation and moments by analogy to bivariate. Some functions for working with multivariate samples are available in R, for example, for multivariate normal distributions in the packages **MASS** [178] and **mvtnorm** [67] and for multivariate t-distributions in the latter.

B.3.2 Bernoulli and Binomial Random Variables

The second example of a discrete random variable that we consider is the *Binomial random variable*. We can type `?Binomial` or `?rbinom` to read about its "four functions." A Binomial random variable is part of a parameterized distribution family $Bi(s,p)$, the Binomial family. It has two parameters "size" s, often denoted "n", and "prob," often denoted "p." In the special case where "size" = 1, the Binomial family is just the Bernoulli family with parameter p. One way to generate a single Binomial random variable X of size s with probability p is to generate a sample of s Bernoulli random variables with probability p: $B_1, B_2, \ldots, B_s$. Then set $X = \sum_{i=1}^{s} B_i$. Each of the Bernoulli variables is either 0 or 1 and the sum must be an integer between 0 and s. The resulting Binomial is a discrete random variable with outcomes $\{0, 1, 2, \ldots, s\}$ and probabilities

$$P[X = k] = \binom{s}{k} p^k (1-p)^{s-k}. \tag{B.9}$$

The first term on the left hand is the Binomial coefficient. In R, this result can be obtained more directly using the function `rbinom`, taking arguments, "`n`" for the number of random values to generate, "`size`" for the parameter s and "`prob`" for the probability of a success. Note: for large samples, `rbinom` is considerably faster than adding up many Bernoulli random variables.

B.3.3 Uniform Random Variables

The uniform distribution on (0,1) is part of a location scale family $\mathscr{U}(a,b)$ that comprises distributions uniform on intervals (a,b) with pdf[2]

$$f(x;a,b) = \frac{1}{b-a}, \qquad a < x < b \tag{B.10}$$

[2]It is tedious to write "0 otherwise" as we would have to do for many of the distributions starting with the uniform above. When we define distributions from now on it is understood that the pdfs are 0 at points not specified in the definition and the cdfs are 0 or 1 as appropriate.

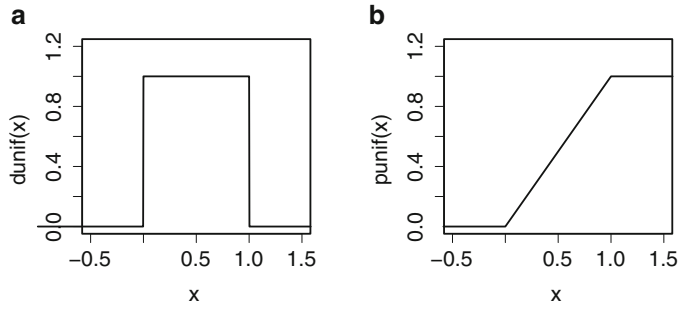

Fig. B.6 The probability density function (pdf) and cumulative distribution function (cdf) of a uniform random variable $\mathscr{U}(0,1)$

and cdf

$$F(x;a,b) = \frac{x-a}{b-a}, \qquad a < x < b. \tag{B.11}$$

The pdf and cdf are plotted in Fig. B.6.

The family of functions `dunif`, `punif`, `qunif`, and `runif` in R allow the user to work with the uniform distribution. The logical expression `runif(size) < p` generates a logical vector of length `size`. Each entry is `TRUE` with probability p and otherwise `FALSE`. The function `sum` treats `TRUE` as 1 and `FALSE` as 0 (a Bernoulli variable $\mathscr{B}(p)$ with 1 replaced by `TRUE`, 0 replaced by `FALSE`) and so B is a number from 0 to `size` distributed as a Binomial random variable. Of course, it is more efficient to use `rbinom` to generate such random values.

Reparameterization

The family $\mathscr{U}(a,b)$ is actually a location-scale family but a,b are not its "natural" location and scale parameters. To put the family in location and scale form we would define new parameters such as its "center" $m = (a+b)/2$ and its "width" $s = (b-a)$.

The $\mathscr{U}(m,s)$ family with this change of parameters represents all the same distributions but now with an obvious location parameter and an obvious scale parameter. This change is an example of reparameterization. We have simply replaced one set of parameters by another using a one-to-one transformation from (a,b) to (m,s). We demonstrate that the transformation is one-to-one by writing down the inverse transformation: $b = m + s/2$ and $a = m - s/2$. The original parameters (a,b) are so useful and easy to work with that you will rarely see this location-scale family parameterized as a location scale family. When we discuss estimation of parameters, the issue of reparameterization will arise again in Sect. B.4.3 below.

B.3.4 Exponential and Gamma Random Variables

The exponential distribution is part of a family known as the Gamma family with parameters n and λ. When $n = 1$, the Gamma random variable is also called the *exponential random variable*. An exponential random variable has pdf

$$f(x;\lambda) = \lambda e^{-\lambda x}, \qquad x > 0 \tag{B.12}$$

and cdf

$$F(x;\lambda) = 1 - e^{-\lambda x}, \qquad x > 0. \tag{B.13}$$

The exponential is widely used to model random delays in time and the equation above often appears with t (denoting time) instead of x as the argument. A very common reparameterization of the exponential is to replace λ by $\tau = \lambda^{-1}$. The exponential is often used to model the completion time of decision processes in cognition and is closely connected to the study of reaction times (response times). If we think of the occurrence of the exponential random variable as an "event" in time, we can assign units of "mean events per unit time" to λ and "mean time to event" to τ.

With reparameterization, the exponential distribution is a scale family with scale parameter τ and it could be made into a location-scale family by adding a location parameter.

The Gamma family $\Gamma(n,\tau)$ contains random variables that are the sums of n independent, identically distributed exponential random variables with rate parameter τ. Its pdf is

$$\gamma(x;n,\tau) = C_n \tau^n x^{n-1} e^{-x/\tau}, \qquad x > 0. \tag{B.14}$$

The constant C_n is chosen so that the area under the pdf is 1 ($C_1 = 1$). When $n = 1$ the pdf of the Gamma is the pdf of the exponential with parameterization in terms of τ. Its cdf does not have a closed form except when $n = 1$.

Some examples of the pdf for different values of τ are given in Fig. B.7 The parameter n is an example of a *shape parameter*. When n increases the shape of the Gamma pdf changes and the change is not just a simple change in location or scale. The parameter n of the binomial $\mathscr{B}(x;n,p)$ is also a shape parameter.

A sample from the Gamma distribution is shown in Fig. B.1b above to simulate human response times. The Gamma function has also been used to model the duration of the states of bi-stable figures [124].

B.3.5 The Exponential Family

The *exponential family* [109,127, Sect. 1.5] is a broad class of distributional families all of whose pdfs can be written in the form

$$f(x|\theta) = h(x)g(\theta)e^{\eta(\theta)T(x)}, \tag{B.15}$$

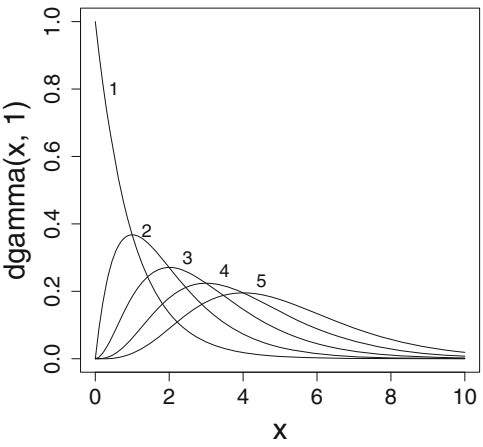

Fig. B.7 The probability density function (pdf), $\gamma(x; n, \tau)$, of a Gamma random variable, $\Gamma(n, \tau)$, for various values of n and $\tau = 1$

where y is a random variable, and the θ are unknown parameters to be estimated. For the Gaussian distribution, for example $\theta = (\mu, \sigma^2)$. and h, g, η, and T are known functions [53]. The choice of functions h, g, η, and T specifies a particular parametric family of distributions with parameters θ such as for the Gaussian. Any distributional family whose pdf can be written in the form of the equation above is part of the exponential family. Many commonly used distributional families including the Gaussian, exponential, Gamma, χ^2, Beta, Bernoulli, binomial, and Poisson are part of the exponential family. The t-distribution is an example of a distribution that is *not* in the exponential family. The exponential family is sometimes referred to as the *exponential class* but the term "exponential family" is far more common. It is doubly unfortunate since its members are themselves distributional families and the exponential distributional family (Sect. B.3.4) plays no special role. It is just another exponential family distribution.

The equation above is often presented in alternative forms. If, for example, we set $A(\theta) = \log(g(\theta))$ and $B(x) = \log(h(x))$, then the equation above becomes

$$f(x|\theta) = \mathrm{e}^{\eta(\theta)T(x)+A(\theta)+B(x)}. \tag{B.16}$$

The change in form is purely cosmetic. It is sometime written including an additional dispersion parameter, ϕ [127, 178], but this will be assumed to be known in nearly every case that we consider here.

Exponential family distributions have desirable statistical properties that go beyond the scope of this book. See [109, Sect. 1.5].

B.3.6 Monte Carlo Methods

The values generated by R functions such as `rbinom` and `rnorm` are not random. They are the output of algorithms that generate deterministic sequences of numbers that mimic many properties of random variables including their unpredictability. The original versions of these algorithms were inspired by the needs of scientists working on the Manhattan Project during the Second World War and were nicknamed "Monte Carlo" methods after the eponymous country and its famous casino. The key insight is that certain computations produce sequences of numbers that have many of the properties of random sequences while being completely deterministic. The sequence $0,0,1,0,0,0,1,1,0,0,0,1,\ldots$ would pass most statistical tests for a Bernoulli random variable with $p = 0.5$ even though it is just the digits of π to base 10, classified as even (1) or odd (0). Designing such "pseudo-random number generators (pRNGs)" is difficult and considerable work has gone into developing pRNGs that are useful for simulation and data analysis [20, 52]. Knuth describes his amusing attempts to develop a "super random" pRNG and how it failed [100, Chap. 3].

Current computer languages (including, of course, R) have Monte Carlo procedures that have been carefully tested. Although they are all deterministic algorithms, they can be treated as random in modeling physical (including biological) stochastic processes. To learn about R's facilities for random number generation and how to modify them, see `?RNG`.

B.4 Estimation

Suppose that we are given a sample $X_1,\ldots,X_n$ from a random variable in a parametric family with parameters $\theta_1,\ldots,\theta_m$. For example, we have data from the random variable in the Gamma family but we do not know the value of its shape parameter n and its scale parameter τ. We want to estimate these "unknown parameters" from the sample of data. For simplicity, let's focus on the case where the distributional family has only one parameter θ, i.e., $m = 1$. An *estimator* of θ is any function of the data

$$\hat{\theta} = T_{\theta_i}(X_1,\ldots,X_n). \tag{B.17}$$

The first time you encounter this definition it may seem odd. We emphasize that any function of the data can be termed an estimator (also known as a *statistic* or *statistical estimator*). Even apparently pathological estimators such as $\hat{\theta} = T_{\theta_i}(X_1,\ldots,X_n) = 7$. This is the estimator that throws away the data and always estimates that θ is 7. This quirk in statistical terminology is intentional. It is easier to discuss the advantages and disadvantages of estimators if we first cast a broad net.

In effect, from among all of the estimators possible, we would like to find estimators that are good. And before doing so, we need to define what we consider to be good.

B.4.1 What Makes an Estimate "Good"?

There are multiple competing criteria for goodness in estimation, and consideration of different criteria and how to design statistics that achieve them would make up a rigorous course in mathematical statistics. Here we consider two of the most venerable, unbiasedness and minimum variance. An estimator is just a function of the random variables in the sample and is itself a random variable. We can consider its expected value for any specific value of θ:

$$\mathsf{E}_\theta[T_\theta] = \int\int\cdots\int T_\theta(x_1,\ldots,x_n) f(x_1;\theta) f(x_2;\theta)\ldots f(x_n;\theta)\,\mathrm{d}\theta. \tag{B.18}$$

Note that this expected value is not a random variable. We integrated out all of the randomness in taking the expected values.

An estimator is *unbiased* if and only if $\theta = E_\theta[T_\theta]$. In words, a statistic is unbiased if and only if it is "on average" equal to what it is supposed to be estimating.

A second criterion uses the variance of an estimator.

$$Var_\theta[T_\theta] = \int\int\cdots\int (T_\theta(X_1,\ldots,X_n) - E_\theta[T_\theta])^2 f(x_1;\theta) f(x_2;\theta)\ldots f(x_n;\theta)\,\mathrm{d}\theta. \tag{B.19}$$

We can now cut down the universe of all possible estimators by focusing on estimators that are unbiased and that have very low variance for all possible estimates of θ. The last clause "...all possible estimates of θ" is important. Otherwise, the "silly estimator" $T_\theta(X_1,\ldots,X_n) = 7$ looks really good when $\theta = 7$ (zero variance!) but less so for other values of θ.

One particular class of estimators in common use are *uniform minimum variance unbiased estimators (UMVUE)*. These are estimators that for every possible value of θ are unbiased and, again for every possible value of θ have a variance no greater than that of any other unbiased estimator. For a particular problem, there need be no UMVUE and for many problems there is no easy way to prove that a particular estimator is or is not the UMVUE. An example of a UMVUE: suppose that we have a sample $X_1,\ldots,X_n$ from a normal distribution $\mathcal{N}(\mu,\sigma^2)$, and we want to estimate the location parameter μ. Then the mean of the sample

$$\hat{\mu} = \bar{X} = (X_1 + \cdots + X_n)/n \tag{B.20}$$

is an UMVUE. Proving this is beyond the scope of this appendix. You can consult any book on mathematical statistics (e.g., [129]).

B.4.2 Maximum Likelihood Estimation

We will use a particular estimation method to derive many estimators in the applications presented in this book. There are other possible choices of methods for devising estimators and we will discuss one, *Bayesian estimation*, very briefly in a later section of this Appendix. Maximum likelihood estimators are easy to derive, and they can usually be computed numerically using numerical optimization methods, such as `optim` in R. Also, for many well-known problems, the resulting estimators are standard, i.e., the ones already in common use.

Suppose we have a sample of data $X_1, \ldots, X_n$ from a parametric distribution with pdf $f(x; \theta_1, \ldots, \theta_m)$. We compute the *likelihood* of the sample as,

$$\mathscr{L}(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n) = \prod_{i=1}^{n} f(X_i; \theta_1, \ldots, \theta_m). \tag{B.21}$$

Two observations: First, the *variables* in the likelihood function are the unknown parameters $\theta_1, \ldots, \theta_m$. The sample, after we have collected the data, is just a collection of numbers that are fixed, the data we just collected. For that reason, we write the likelihood function as $\mathscr{L}(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n)$, emphasizing that the focus of attention is now on the unknown values of the parameters $\theta_1, \ldots, \theta_m$. Second, if the random variable X is discrete, then the pdf $f(x; \theta_1, \ldots, \theta_m)$ is actually a pmf (see Sect. B.2.4) and the likelihood function is simply the probability that the observed sample $X_1, \ldots, X_n$ would occur if the parameters had the specific values $\theta_1, \ldots, \theta_m$. If, however, the random variable is continuous, the likelihood function cannot be interpreted as a probability. Indeed, it is not even a pdf in its own right since if we integrated over the parameters $\theta_1, \ldots, \theta_m$ the result is very unlikely to be 1. The likelihood function is the likelihood function: we are interested in "mining" it for information about the parameters.

There is evidently some information about the parameters in the likelihood function. If, for example, we are considering the uniform family $\mathscr{U}(a, b)$ with unknown parameters a and b and one of our sample values $X_1 = 2.31$, then it is clear that $a < 2.31 < b$. Otherwise, the sample value could not have occurred.

To turn the likelihood function (B.21) into an estimation method, we simply decree that our estimators of the unknown parameters $\theta_1, \ldots, \theta_m$ are the values $\hat{\theta}_1, \ldots, \hat{\theta}_m$ that maximize the likelihood function. The maximum is usually unique, but if it is not then we have multiple choices of estimators. This almost never occurs in practice. We will typically reserve the notation $\hat{\theta}$ for the maximum likelihood estimator of θ.

Example

Suppose that we take a sample $X_1, \ldots, X_n$ from a normal distribution with pdf

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{B.22}$$

and compute the likelihood function

$$\mathscr{L}(\mu,\sigma;X_1,\ldots,X_n)=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(X_i-\mu)^2}{2\sigma^2}} \tag{B.23}$$

It is scarcely obvious that we can maximize this likelihood equation by choice of μ and σ using calculus and get closed form solutions,

$$\hat{\mu}=\bar{X}=\frac{\sum_{i=1}^{n}X_i}{n} \tag{B.24}$$

and

$$\hat{\sigma}=\sqrt{\frac{\sum_{i=1}^{n}(X_i-\bar{X})^2}{n}} \tag{B.25}$$

The formula for $\hat{\mu}$ is likely familiar while that for $\hat{\sigma}$ may look a bit odd with n in the denominator instead of $n-1$. We will return to this point later.

Instead of maximizing likelihood we can instead maximize the logarithm to any base of likelihood by choice of $\hat{\theta}_1,\ldots,\hat{\theta}_m$. If the value $\hat{\theta}$ maximizes $\mathscr{L}(\theta_1,\ldots,\theta_m;X_1,\ldots,X_n)$, then it also maximizes $\log\mathscr{L}(\theta_1,\ldots,\theta_m;X_1,\ldots,X_n)$. We can take the logarithm since the likelihood function is nonnegative (we allow for $\log(0)=-\infty$ and treat it as any other number). The log likelihood function

$$\log\mathscr{L}(\theta_1,\ldots,\theta_m;X_1,\ldots,X_n)=\sum_{i=1}^{n}\log f(X_i;\theta_1,\ldots,\theta_m) \tag{B.26}$$

is often much simpler to write down and sometimes we can maximize it by simple calculus. To return to the normal example above,

$$\log\mathscr{L}(\mu,\sigma;X_1,\ldots,X_n)=-\sum_{i=1}^{n}\frac{(X_i-\mu)^2}{2\sigma^2}-n\log\sigma+C \tag{B.27}$$

where the constant term C contains the contributions of $\sqrt{2\pi}$. We are only planning to maximize likelihood or its logarithm and any constant terms do not affect maximization. They can be neglected and often we will leave them out altogether. The resulting log likelihood function for the normal sample is much simpler than the likelihood function and we can see clearly that it is written in terms of a sum of squared differences. Maximizing log likelihood for the normal will prove to be equivalent to minimizing the sum of squared differences and the well-known connection between normal estimation and so-called "least squares" emerges from the log likelihood equation. Moreover, the resulting ML estimate $\hat{\mu}$ of μ is UMVUE and the "MV" part is a bit more plausible given the connection to minimizing least squares.

Example

A second example of an ML estimate is that for a Bernoulli sample $B_1, \ldots, B_n$ from a $\mathscr{B}(p)$ distribution. Here there is only one parameter $m = 1$. The likelihood function is

$$\mathscr{L}(p; B_1, \ldots, B_n) = \prod_{i=1} n p^{B_i} (1-p)^{1-B_i} \tag{B.28}$$

and the log likelihood function is

$$\log \mathscr{L}(p; B_1, \ldots, B_n) = \sum_{i=1} n(B_i \log(p) + (1-B_i)\log(1-p)). \tag{B.29}$$

We can solve for the maximum of either by differentiating with respect to p. For the log likelihood function the derivative is

$$\frac{\mathrm{d}}{\mathrm{d}p} \log \mathscr{L}(p; B_1, \ldots, B_n) = \sum_{i=1} n \left(\frac{B_i}{p} - \frac{1-B_i}{1-p} \right) \tag{B.30}$$

which we set to 0 and solve for the maximum $\hat{p} = (\sum_{i=1}^{n} B_i / n)$, the proportion of Bernoulli outcomes that are 1. This outcome is scarcely surprising but certainly comforting. The intuitively obvious estimate is the MLE.

In log likelihood form we can see that each value in the sample contributes a "vote" to the log likelihood function of the form $\log f(X_i; \theta_1, \ldots, \theta_m)$ and that the overall log likelihood function is the sum of these votes. We can denote these separate contributions as L_i, the summands of the log likelihood function. In the normal case, each value in the sample contributes an inverted parabola

$$\log \mathscr{L}_i(\mu, \sigma : X_i) = -\frac{(X_i - \mu)^2}{2\sigma^2} - \log(\sigma) \tag{B.31}$$

and the value that maximizes $\log \mathscr{L}_i$ is obviously X_i. We show the parabolas for a sample of size 3 in Fig. B.8. The overall log likelihood function for $\sigma = 1$ is also an inverted parabola which is the sum of the $\log L_i$ as shown in bold in Fig. B.8.

Of course, the log likelihood function of the normal depends on both μ and σ. If we plot these functions for other values of σ the height of the parabolas and their width will change. But the basic picture remains. Each X_i is voting for values near X_i and the overall estimate (the thicker black curve) is a combination of these "votes." Notice that it is narrower. The accumulated support from the three sample values is converging on an estimate. The log likelihood function is sometimes referred to as the *support function* and it is clear from the figure how the data "support" different possible estimates.

If the likelihood function is a probability (the random variable X is discrete as in the Bernoulli example), then we are selecting the values of the parameters that are most likely to have given rise to the observed sample. Stated in this form, it is not obvious why one would employ such a criterion. Unlike criteria such as

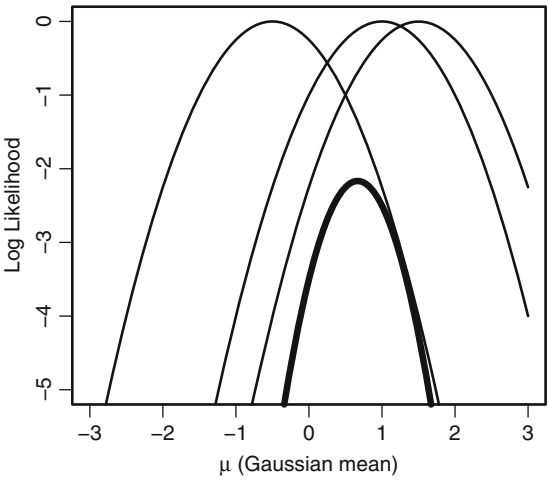

Fig. B.8 The support functions for each entry in a sample of size 3 of a Gaussian random variable with $\sigma = 1$ and the sum of these support functions into an overall support function

"minimum variance," it is not obvious that a maximum likelihood estimate has any desirable property per se. In the next section we will consider the properties that result from using maximum likelihood estimation. In any case, if the random variable X is continuous, then the likelihood function is not a probability or even a pdf, as mentioned above, and it is even less obvious why one would choose to maximize likelihood.

B.4.3 Why Likelihood? Some Properties

Asymptotically UMVUE

ML estimates are, in general, *not* unbiased and *not* minimum variance. However, as the sample size increases, they converge to estimates that are unbiased and minimum variance. We say that ML estimators are asymptotically UMVUE. Of course, for any specific application, the ML estimates may still exhibit appreciable bias and it is little consolation to know that, in a similar application with a much larger data set, the bias would be reduced. In practice then it is always appropriate to verify that estimator bias is small enough to be neglected.[3]

[3]One reaction to such promises of asymptotic good performance is that of J. M. Keynes, "In the long run we are all dead." Still, for particular examples such as estimating the parameter μ of the normal, the ML estimator may be not only unbiased but also an UMVUE. Life is sometimes good.

Invariance Under Reparameterization

An important advantage of ML estimators is that they are *invariant under reparameterization*. The exact choice of parameters for any parametric family is arbitrary. A change in parameters is called *reparameterization* (see Sect. B.3.3). In considering ML estimation of parameters of the normal, we chose to estimate the parameter σ. Of course, we could have chosen to estimate the parameter $V = \sigma^2$. The former is the standard deviation of the normal distribution and is a scale parameter; the latter is its variance. Both parameterizations can be argued for and it is interesting that the normal is often denoted $\mathcal{N}(\mu, \sigma^2)$, effectively preserving both parameterizations! At first glance the choice of parameterization would seem to be mainly a matter of taste: "you say tomayto, I say tomahto." However, suppose that you have chosen σ as parameterization and estimated it as $\hat{\sigma}$ using maximum likelihood. We have taken the same sample and estimated the variance $\hat{\sigma^2}$. If your estimated squared $(\hat{\sigma})^2$ is not equal to our estimate, then we have a conflict. Our estimates should be consistent with one another but they disagree. The inconsistency is serious, as serious as getting different estimates of the location of a planet if we calculated in miles or in kilometers.[4] If, however, we both used ML estimates, then there can be no such disagreement: $(\hat{\sigma})^2 = \hat{\sigma^2}$ always. This is invariance under reparameterization and it always holds if we use ML estimators. Invariance under reparameterization can fail if we use Bayesian estimation methods (See below, Sect. B.4.4).

In an elementary introduction to estimation, we would normally not discuss invariance under reparameterization. However, in considering the psychophysical methods in several of the chapters, it will prove to be a very useful tool. If we can, for example, change parameterization and make a difficult estimation problem easy, then we will do so, knowing that within the framework of ML estimation, we are entitled to do so.

Example

We noted that the formula for the MLE of σ given a sample from a normal distribution was

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}} \tag{B.32}$$

and we can therefore write down the MLE of the variance parameter σ^2 as

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}. \tag{B.33}$$

[4] Which NASA once did, leading to the loss of the Mars Climate Orbiter spacecraft in 1999.

The reader is likely familiar with the alternative estimator

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1} \tag{B.34}$$

available on most hand calculators. The reason that the alternative is preferred is that it is unbiased $E[S^2] = \sigma^2$ and therefore the ML estimator is biased $E[\hat{\sigma}^2] = \frac{n-1}{n}\sigma^2$. We can see that as the sample size increases, $n-1/n \to 1$ and the bias vanishes asymptotically.

While the ML estimator may be biased we can often correct for the bias either in closed form as in the example just given or by means of simulation using resampling methods to estimate bias. See [55].

B.4.4 Bayesian Estimation

An alternative to MLE is Bayesian estimation. The estimation problem is unchanged. We have a sample of data $X_1, \ldots, X_n$ from a parametric distribution with pdf $f(x; \theta_1, \ldots, \theta_m)$. However, now we assume that the true values of the parameters $\theta_1, \ldots, \theta_m$ are the realization of a random variable $\Theta = (\theta_1, \ldots, \theta_m)$ with its own pdf $\pi(\theta_1, \ldots, \theta_m)$ referred to as the *prior distribution*. We would like to use the prior together with the sample $X_1, \ldots, X_n$ to improve our estimates of $\theta_1, \ldots, \theta_m$. Of course, to do so, we need to know what the prior is.

Suppose, for example, that we are trying to estimate the peak sensitivity, λ_{L}, of the long-wavelength (L) photoreceptor of a specific individual. We collect data in a psychophysical task and we could use an ML method to estimate λ_{L}. In doing so, we use only the data and the assumptions made in modeling the psychophysical task. But suppose that we know that the individual was drawn "at random" from a particular population and that we have estimates of λ_{L} for many other individuals drawn from the same population as shown in Fig. B.9. These estimates are not a prior (they are just a histogram) but we might be willing to use them to estimate a distribution (shown as a solid curve) that we will treat as a prior. Because of the prior, we know something about λ_{L} even before we collect the data.

The first step in computing a Bayesian estimate is the same as in computing the ML estimate. We compute the likelihood of the sample,

$$\mathscr{L}(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n) = \prod_{i=1}^{n} f(X_i; \theta_1, \ldots, \theta_m) \tag{B.35}$$

and then we multiply it by the prior $\pi(\theta_1, \ldots, \theta_m)$. The result need not be a probability density function (it may not integrate to 1), but we can normalize it to get the *posterior distribution*

$$p(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n) = C^{-1} L(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n)\pi(\theta_1, \ldots, \theta_m). \tag{B.36}$$

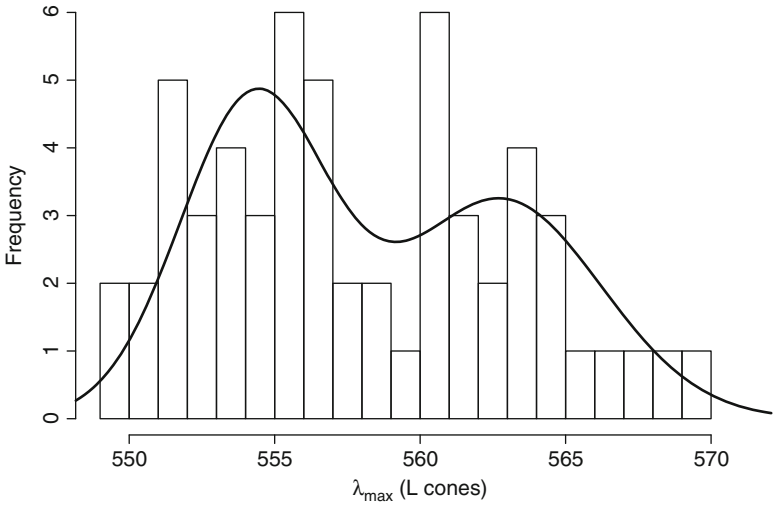

Fig. B.9 Histogram of peak wavelengths ($\lambda_{\max}$) for the human L-cone photopigment as measured using microspectrophotometry of cone photoreceptors [45]. The *solid curve* is a best-fitting sum of Gaussians

The normalizing constant is just

$$C = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n)\, \mathrm{d}\theta_1, \ldots, \theta_m. \tag{B.37}$$

Equation (B.36) is a form of Bayes Theorem applied to probability density functions. We interpret the equation as a method for taking the prior (which contains all the information we have about the distribution before the data are collected), and combining it with the data, summarized by the likelihood function to get a new version of the prior, the posterior. We interpret the posterior as our best estimate of the pdf of the parameters, and, thereafter, we can derive estimates of the parameters from it. For example, by analogy to maximizing likelihood, the values of the parameters that maximize the posterior are the *maximum a posteriori* (MAP) *estimate* of the parameters. The MAP is just the mode of the posterior distribution. However, since we interpret the posterior $p(\theta_1, \ldots, \theta_m; X_1, \ldots, X_n)$ as a probability density function we can also consider other measures such as the mean of the posterior or its median as alternatives.

The key differences between the ML approach and Bayesian methods are typically not very great in practice. The key conceptual difference is the assumption that the parameters we seek to estimate are themselves random variables and that an intermediate goal of estimation is to estimate the distribution of the parameters given all available knowledge including the prior. As just presented, it would be difficult to criticize the Bayesian approach and, in fact, there is little controversy

surrounding this case (referred to as *empirical Bayesian estimation*) where the parameters are readily interpretable and we do know their prior distribution exactly or approximately.

Some of the approaches used in this book fit data by weighting the likelihood by a penalty, which is formally equivalent to the procedure described above for introducing a prior. For example, GAM or generalized additive models penalize the likelihood by a function that limits how "wiggly" the function fit to the data will be and, thus, uses a prior that prefers a smoother function to fit the data than one that would go through every point [198]. The random effects terms of mixed-effects models also correspond to a penalization of the likelihood [141]. The penalty in this case serves to limit the complexity of the model, not unlike the goal of the smoothing for GAMs. In both of these cases, the presence of the penalty serves to produce fits that balance a model's fidelity to the data and its complexity. Thus, the prior is to prefer less complex models.

Somewhat more controversial is the application of Bayesian methods where the parameters are not readily interpretable as random draws from a population or where the prior is unknown and must be assumed. We will return to this point below.

Example

We wish to estimate the probability that a particular coin will land "heads" when tossed. As in Sect. B.4.2, we will model the coin as a Bernoulli random variable $\mathscr{B}(p)$. Suppose that we toss the coin ten times and obtain "heads" three times and "tails" seven times. We saw above in Sect. B.4.2 that the MLE of p was $\hat{p} = 0.3$, the proportion of "heads." But now suppose that the coin is drawn from a large population of coins whose probabilities of landing "heads" are unknown but may vary. We denote the probability that our particular coin lands "heads" by P, following our usual convention that random variables are denoted by uppercase letters. P is a random variable since in drawing the coin from a population of coins we have effectively drawn P from all populations of possible probabilities. To apply Bayesian estimation we need a prior distribution for P. Let's begin by assuming that we are completely ignorant of how coins behave and to represent our ignorance we set the pdf of P to be $\mathscr{U}(0,1)$, the uniform distribution. Suppose that we toss the coin n times, modeling the outcomes as Bernoulli random variables, $B_1, B_2, \ldots, B_n$ and apply (B.36).

We find that the posterior

$$p(P; B_1, \ldots, B_n) = C \prod_{i=1} n p^{B_i} (1-p)^{1-B_i} \tag{B.38}$$

is just the likelihood function multiplied by a positive constant (the constant needed to insure that the posterior is a pdf). The resulting distribution is an example of a *Beta distribution*.

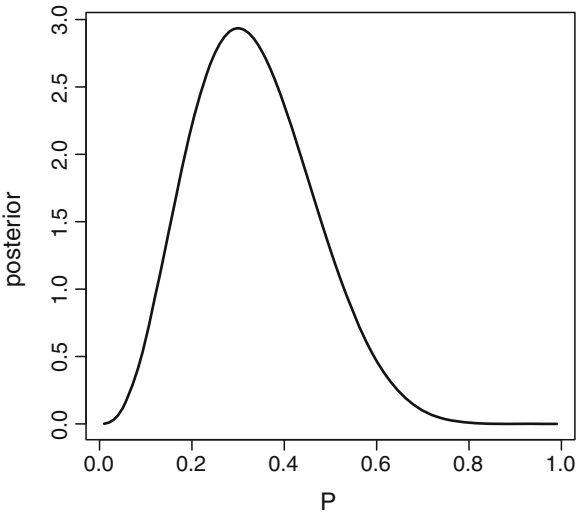

Fig. B.10 The posterior distribution of P, the parameter of a Bernoulli $\mathscr{B}(P)$ random variable following a sample with three 1s and seven 0s. The prior distribution was $\mathscr{U}(0,1)$. The resulting distribution is a Beta distribution

The Beta family has two parameters α, β The pdf of a Beta distribution is 0 outside of the interval $(0,1)$ and is of the form

$$f(x;\alpha,\beta) = p^{\alpha-1}(1-p)^{\beta-1} \qquad p \in (0,1). \tag{B.39}$$

In our example, α is one more than the number of heads, and β is one more than the number of tails. With three "heads" and seven "tails," the resulting posterior is shown in Fig. B.10. With a uniform prior, the MLE and MAP are identical (the positive constant cannot alter the location of the maximum), and they are both the proportion of "heads," 0.3. However, now we can interpret the likelihood/posterior as a pdf and use it to estimate the probability that $P < 0.5$, among other things.

Example

Suppose that we are uncomfortable with the assumption that the prior of P is uniform. After all, most coins we have encountered have probabilities p close to 0.5. Then we might assume a different prior. One clever way to do that is to pick a prior that when multiplied by the likelihood results in a posterior distribution that is easy to work with. We can, for example, pick the prior to be a Beta distribution with parameters $\alpha = 3$ and $\beta = 3$ or $\alpha = 5$ and $\beta = 5$ or $\alpha = 10$ and $\beta = 10$. The dotted and dashed curves in Fig. B.11 are the resulting priors, the broader dotted curve that for $\alpha = 3$ and $\beta = 3$ and the narrower dotted curve being that for $\alpha = 10$ and $\beta = 10$. All three are symmetric around 0.5. These are intended to capture our prior beliefs concerning the coin and we pick the middle one $\alpha = 5$ and $\beta = 5$ (dashed). We then toss the coin ten times and get three "heads" and seven "tails." We compute the posterior now incorporating our new prior and the resulting posterior is shown

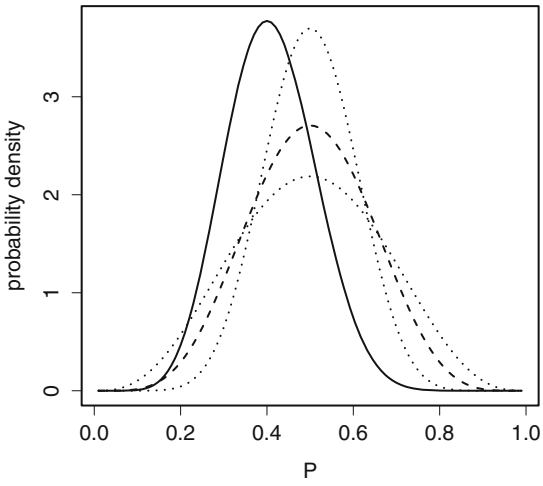

Fig. B.11 Three candidate prior distributions and the posterior distribution of P, the parameter of a Bernoulli $\mathscr{B}(P)$ random variable following a sample with three 1s and seven 0s

as a solid curve in Fig. B.11. The posterior is very different from that obtained with a uniform prior and it is a Beta distribution with parameters $\alpha = 3+5+1$. and $\beta = 7+5+1$ and the MAP is $0.4 = 8/20$. The new prior has led to an estimate that is closer to 0.5 than the estimate obtained with the uniform prior. One controversial aspect of the Bayesian approach is illustrated in these last two examples. The prior is intended to capture the aspects of our knowledge concerning the parameters outside of the sample. When, however, our knowledge is imprecise, it is not obvious that introducing it as a prior produces a better estimate. If, however, the prior distribution had been based on extensive analyses of the probabilities of actual coins drawn from an actual population, then the use of the prior to improve the estimate would generate little controversy.

A more subtle difficulty arises when we engage in Bayesian estimation using the uniform distribution to represent ignorance. If the parameter P of the Bernoulli is drawn from a uniform distribution $\mathscr{U}(0,1)$, then the reparameterization $R = 1/P$ is not. Indeed, the cdf of R

$$P[R < x] = P[1/P < x] = P[P > 1/x] = 1 - 1/x, \qquad 1 \leq x \leq \infty \tag{B.40}$$

and the pdf is x^{-2} over the interval $(1,\infty)$ and otherwise 0. The resulting distribution is not uniform and there is, of course, no uniform distribution over the interval $1 \leq x \leq \infty$. We seem to be simultaneously ignorant about P and quite knowledgeable about its reciprocal.

A similar paradox arises when we consider Bayesian estimation and reparameterization. Suppose we collect a sample from the Bernoulli $\mathscr{B}(P)$ and form a maximum likelihood estimate $\hat{P}$. As a consequence of invariance of reparameterization, the MLE of R is just $1/\hat{p}$. However, the Bayesian estimate with *any* prior on P will not be invariant under reparameterization. An example and discussion can be found in [54, pp. 20–22].

B.5 Nested Hypothesis Testing

Hypothesis testing is a major division of mathematical statistics comparable to estimation and in the form of null hypothesis testing it is mired in controversy. We will avoid controversy by concentrating on the applications of hypothesis testing that are most common in psychophysics. These typically concern a comparison of two parametric models, one of which is a special case of the other. We will characterize two such models by their likelihood functions $\mathscr{L}_1(\theta_1,\ldots,\theta_m,\theta_{m+1},\ldots,\theta_{m+p};X_1,\ldots,X_n)$ and $\mathscr{L}_0(\theta_1,\ldots,\theta_m,\theta_{m+1} = C_1,\ldots,\theta_{m+p} = C_p;X_1,\ldots,X_n)$. The first model has a likelihood function with $m+p$ parameters and the second model is identical to the first except that p of the parameters have been set to specific constant values, $C_1,\ldots,C_p$. The notation above is cumbersome and an example will help.

Example

Suppose that the first model corresponds to the normal family $\mathscr{N}(\mu,\sigma^2)$ with two parameters, μ and σ. The second model is a subset of the distributions in the first family, namely those for which $\mu = 0$. The two likelihood functions are $\mathscr{L}_1(\mu,\sigma;X_1,\ldots,X_n)$, given above in (B.23) and $\mathscr{L}_0(\mu = 0,\sigma;X_1,\ldots,X_n)$ which is the same equation but with every occurrence of μ set to 0.

The maximum likelihood estimates $\hat{\mu}$ and $\hat{\sigma}$ are the values that maximize L_1 and therefore $\mathscr{L}_1(\hat{\mu},\hat{\sigma};X_1,\ldots,X_n)$ is the maximum value of likelihood achievable. Now, we set $\mu = 0$ in $\mathscr{L}_0$ and maximize its likelihood by varying the one remaining parameter σ. The resulting maximum likelihood estimate $\hat{\sigma}_0$ need not be the same as the previous ML estimate $\hat{\sigma}$. It almost certainly won't be the same since the latter is the maximum likelihood when both μ and σ are allowed to vary, and the former corresponds to the maximum when $\mu = 0$ by fiat.

The constrained ML cannot be bigger than the unconstrained:

$$\mathscr{L}_1(\hat{\mu},\hat{\sigma};X_1,\ldots,X_n) \geq \mathscr{L}_0(\mu = 0,\hat{\sigma}_0;X_1,\ldots,X_n) \tag{B.41}$$

If it were, then setting $\mu = 0$ and $\sigma = \hat{\sigma}_0$ would lead to a likelihood value for L_1 greater than that achieved by the maximum likelihood values, an impossibility.

The question we address is, should we reject the constrained model whose likelihood is L_0 in favor of the unconstrained L_1? We interpret the ratio of the likelihoods at their respective maxima as a measure of the improvement in accounting for the data,

$$\Lambda = \frac{\mathscr{L}_1(\hat{\mu},\hat{\sigma};X_1,\ldots,X_n)}{\mathscr{L}_0(\mu,0,\hat{\sigma}_0;X_1,\ldots,X_n)}. \tag{B.42}$$

If this value is large enough, we would be justified in rejecting the constrained model.

To develop a sense of how the resulting *likelihood ratio* statistic behaves, we can make use of a remarkable but asymptotic result known as Wilks' Theorem [193] that tells us that if the sample was in fact drawn from the constrained model/family (in our example, $\mu = 0$ in reality), then the statistic

$$W = 2\log\Lambda \tag{B.43}$$

will be distributed as a χ^2 random variable whose degrees of freedom (df) are the number of parameters constrained (the difference in number of parameters in the two models). In our example, only μ was constrained and the corresponding df is 1. (See [129, p. 441ff].) We would expect the W value to be distributed as a χ^2 variable with one degree of freedom χ^2_1. This result is remarkable since it holds true for any two nested models. For example, let's consider a nested hypothesis test based on the exponential distribution. The unconstrained hypothesis is that the data are drawn from an exponential distribution with parameter $\lambda > 0$. The constrained hypothesis is that $\lambda = 1$ We can verify Wilks' Theorem by drawing a sample of size 100 from an exponential distribution with $\lambda = 1$ and computing the maximum constrained and maximum unconstrained likelihoods. It is computationally convenient to compute the log likelihoods of the two models (to avoid exceeding the numerical precision) and taking their difference. The log likelihood of the sample is

$$\log\mathscr{L}(\lambda; X_1, \ldots, X_n) = -\lambda \sum_{i=1}^{n} X_i + n\log\lambda. \tag{B.44}$$

Given a sample $X_1, X_2, \ldots, X_n$ the unconstrained maximum likelihood estimate of $\lambda = 1/\bar{X}$, and this is the value we use to maximize the unconstrained likelihood. In computing the constrained likelihood we use the value $\lambda = 1$. We repeat this simulation 1,000 times and plot the histogram of the resulting W values with the χ^2_1 pdf superimposed in Fig. B.12. As one should begin to expect (we hope), we obtain the smooth pdf curve using the `dchisq` function. The match illustrates Wilks' Theorem and also provides assurance that the asymptotic distribution is a good model for the actual distribution of W for such a large sample.

Following the logic of hypothesis testing we would pick a size α for our test of the unconstrained model against the constrained and reject if the test statistic W falls above the $1-\alpha$ quantile of the χ^2_1 distribution. In the general case, with p constrained parameters, we would replace χ^2_1 by χ^2_p.

In the discussion above, we used Monte Carlo methods to good effect. Given any asymptotic claim such as Wilks' Theorem, we typically have no guarantee that the sample size in any particular application is "large enough" to be "nearly asymptotic." However,we can simply simulate our experiment with the design we used or plan to use and check how a statistic such as W behaves.

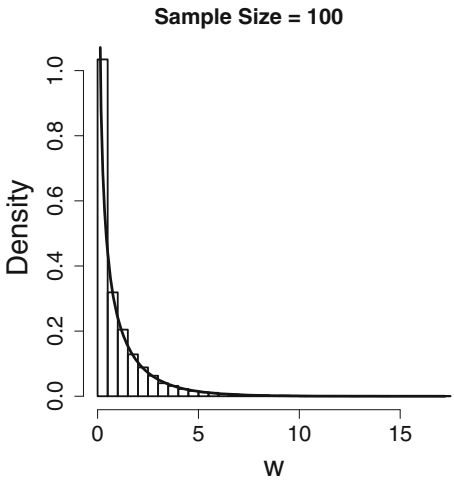

Fig. B.12 Simulation of distribution of Wilk's likelihood ratio statistic. The *solid lines* correspond to the pdf of χ^2_1

B.5.1 Reparameterization: Nested Hypothesis Tests

One very common form of the nested hypothesis test makes use of constraints that do not involve setting parameters to specific values but instead considering that in a list of parameters some of the parameters are identical. That is, in the list $\theta_1, \ldots, \theta_p$ we wish to test whether $\theta_{p-1} = \theta_p$. We can easily recast this problem as simply a reparameterization. We replace the parameters $\theta_1, \ldots, \theta_p$ by the reparameterization $\theta_1, \ldots, \theta_{p-1}, \Delta\theta$ where each θ_i in one parameterization is equal to θ_i in the other, $i = 1, \ldots, p-1$ and $\theta_p = \theta_{p-1} + \Delta\theta$. the reader can verify that we can transform back and forth from one set of parameters to the other and that the constraint $\theta_{p-1} = \theta_p$ in the first parameterization is just the constraint $\Delta\theta = 0$ in the second. Nested hypothesis tests based on likelihood ratios are also invariant under reparameterization and we can simply use the second parameterization in place of the first.

B.5.2 The Akaike Information Criterion

In fitting a model to data, we hope to characterize as accurately as possible a systematic, underlying component that depends on a set of explanatory variables. If the model describes the data well, then the variation that remains after the fit can be attributed to unsystematic factors that have not or cannot be controlled. The accuracy with which the systematic component is characterized depends on the appropriate choice of model for the data under consideration. The ML procedures attempt to find parameter values for which the model is closest to the data, according to the ML criterion. Adding additional parameters will lead to models with a greater likelihood

and that are a better description of the actual data. The danger in fitting the data too well, however, is that one will describe not only the systematic component but the noise as well. This is called *over-fitting* and leads to models that are poor at predicting new data. Likelihood ratio tests of nested models provide one method for approaching this problem, by excluding models for which the added parameters do not significantly increase the likelihood. They do not necessarily shield the user from over-fitting, however, since a parameter that increases the likelihood by serendipitously accounting for unsystematic variation in the data could be selected.

An (or Akaike's) Information Criterion (AIC) [6] is one of a number of information theoretic measures that have been proposed to address this issue. It is defined as

$$\text{AIC} = -2\log(\mathscr{L}) + 2k, \tag{B.45}$$

where $\mathscr{L}$ is the likelihood and k is the number of parameters estimated in fitting the model. The measure is defined so that better fit models correspond to lower values of AIC. The decrease in negative log likelihood obtained by adding parameters (adding complexity) is penalized by adding back in twice the number of parameters to counterbalance the tendency toward over-fitting. The lowest AIC corresponds to a balance between good fit of the model to the data and model parsimony as indicated by the number of free parameters. The formula for the AIC arises in a derivation of fitting the model to the expected value of the data rather than the data, themselves, and in this way suggests a model that is better at predicting future data than one based just on likelihood [198].

Model fitting functions in R based on likelihood typically provide an AIC value in the output or one that can be assessed through an `AIC` method. Some functions provide an additional measure, the Bayesian Information Criterion (BIC), that employs a harsher penalty for model complexity, based on a penalty of $k\log(n)$ where n is the number of observations.

References

1. Abbey, C.K., Eckstein, M.P.: Classification image analysis: estimation and statistical inference for two-alternative forced-choice experiments. J. Vis. **2**, 66–78 (2002)
2. Ahumada, A.J.: Perceptual classification images from vernier acuity masked by noise. Perception **25** ECVP Abstract Suppl. (1996)
3. Ahumada, A.J.: Classification image weights and internal noise level estimation. J. Vis. **2**(1), 121–131 (2002)
4. Ahumada, A.J., Lovell, J.: Stimulus features in signal detection. J. Acoust. Soc. Am. **49**, 1751–1756 (1971)
5. Ahumada, A.J., Marken, R., Sandusky, A.: Time and frequency analyses of auditory signal detection. J. Acoust. Soc. Am. **57**(2), 385–390 (1975)
6. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Petrov, B.N., Csàki, F. (eds.) Second International Symposium on Inference Theory, pp. 267–281. Akadémia Kiadó, Budapest (1973)
7. Baayen, R.H., Davidson, D.J., Bates, D.M.: Mixed-effects modeling with crossed random effects for subjects and items. J. Mem. Lang. **59**, 390–412 (2008)
8. Bates, D.: Fitting linear mixed models. R News **5**, 27–30 (2005). http://www.r-project.org/doc/Rnews/Rnews_2005-1.pdf
9. Bates, D., Maechler, M., Bolker, B.: lme4: Linear mixed-effects models using S4 classes (2011). R package version 0.999375-42. http://CRAN.R-project.org/package=lme4. Accessed date on 2nd August 2012
10. Bates, D., Maechler, M., Bolker, B.: lme4.0: Linear mixed-effects models using S4 classes (2012). R package version 0.9999-1/r1692. http://R-Forge.R-project.org/projects/lme4/. Accessed date on 2nd August 2012
11. Bates, D.M.: lme4: Mixed-Effects Modeling with R. Springer, New York (in preparation). http://lme4.r-forge.r-project.org/book/
12. Bengtsson, H., Riedy, J.: R.matlab: Read and write of MAT files together with R-to-Matlab connectivity (2011). R package version 1.5.1. http://CRAN.R-project.org/package=R.matlab. Accessed date on 2nd August 2012
13. Bernstein, S.N.: Sur l'ordre de la meilleure approximation des fonctions continues par les polynômes de degré donné. Mémoires de l' Académie Royale de Belgique **4**, 1–104 (1912)
14. Bishop, Y.M.M., Fienberg, S.E., Holland, P.W.: Discrete Multivariate Analysis: Theory and Practice. MIT, Cambridge (1975)
15. Block, H.D., Marschak, J.: Random orderings and stochastic theories of responses. In: Olkin, I., Ghurye, S., Hoeffding, W., Madow, W., Mann, H. (eds.) Contributions to Probability and Statistics, pp. 38–45. Stanford University Press, Stanford (1960)

K. Knoblauch and L.T. Maloney, *Modeling Psychophysical Data in R*, Use R! 32,
DOI 10.1007/978-1-4614-4475-6,

16. Boeck, P.D., Wilson, M. (eds.): Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach. Springer, New York (2004)
17. Boring, E.G.: Sensation and Perception in the History of Experimental Psychology. Irvington Publishers, Inc., New York (1942)
18. Bouet, R., Knoblauch, K.: Perceptual classification of chromatic modulation. Vis. Neurosci. **21**, 283–289 (2004)
19. Bracewell, R.N.: The Fourier Transform and its Applications, 3rd edn. McGraw-Hill, New York (2000)
20. Bratley, P., Fox, B.L., Schrage, L.E.: A Guide to Simulation. Springer, New York (1983)
21. Brindley, G.S.: Two more visual theorems. Q. J. Exp. Psychol. **12**, 110–112 (1960)
22. Britten, K.H., Shadlen, M.N., Newsome, W.T., Movshon, J.A.: The analysis of visual motion: A comparison of neuronal and psychophysical performance. J. Neurosci. **12**(12), 4745–4765 (1992)
23. Broström, G., Holmberg, H.: glmmML: Generalized linear models with clustering (2011). R package version 0.82-1. http://CRAN.R-project.org/package=glmmML. Accessed date on 2nd August 2012
24. Burnham, K.P., Anderson, D.R.: Model Selection and Inference: A Practical Information-Theoretic Approach. Springer, New York (1998)
25. Canty, A., Ripley, B.D.: boot: Bootstrap R (S-Plus) Functions (2012). R package version 1.3-5
26. Carney, T., Tyler, C.W., Watson, A.B., Makous, W., Beutte, B., Chen, C.C., Norcia, A.M., Klein, S.A.: Modelfest: Year one results and plans for future years. In: Rogowitz, B.E., Pappas, T.N. (eds.) Proceedings of SPIE: Human vision and electronic imaging V, vol. 3959, pp. 140–151. SPIE, Bellingham (2000)
27. Carroll, L.: Through the Looking Glass and Ahat Alice Found There. Macmillan, Bassingstoke (1871)
28. Chambers, J.M., Hastie, T.J. (eds.): Statistical Models in S. Chapman and Hall/CRC, Boca Raton (1992)
29. Charrier, C., Maloney, L.T., Cherifi, H., Knoblauch, K.: Maximum likelihood difference scaling of image quality in compression-degraded images. J. Opt. Soc. Am. A **24**, 3418–3426 (2007)
30. Chauvin, A., Worsley, K., Schyns, P., Arguin, M., Gosselin, F.: Accurate statistical tests for smooth classification images. J. Vis. **5**, 659–667 (2005)
31. Christensen, R.H.B.: ordinal—regression models for ordinal data (2010). R package version 2011.09-14. http://www.cran.r-project.org/package=ordinal/. Accessed date on 2nd August 2012
32. Christensen, R.H.B., Brockhoff, P.B.: sensR—an R-package for sensory discrimination (2011). R package version 1.2-13. http://www.cran.r-project.org/package=sensR/. Accessed date on 2nd August 2012
33. Christensen, R.H.B., Hansen, M.K.: binomTools: Performing diagnostics on binomial regression models (2011). R package version 1.0-1. http://CRAN.R-project.org/package=binomTools. Accessed date on 2nd August 2012
34. Chung, K.L., Aitsahlia, F.: Elementary Probability Theory, 4th edn. Springer, New York (2006)
35. Clark, H.H.: The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. J. Verbal Learn. Verbal Behav. **12**, 355–359 (1973)
36. Cleveland, W.S., Diaconis, P., McGill, R.: Variables on scatterplots look more highly correlated when the scales are increased. Science **216**, 1138–1141 (1982)
37. Cleveland, W.S., McGill, R.: Graphical perception: Theory, experimentation and application to the development of graphical methods. J. Am. Stat. Assoc. **79**, 531–554 (1984)
38. Cleveland, W.S., McGill, R.: The many faces of a scatterplot. J. Am. Stat. Assoc. **79**, 807–822 (1984)
39. Conway, J., Eddelbuettel, D., Nishiyama, T., Prayaga, S.K., Tiffin, N.: RPostgreSQL: R interface to the PostgreSQL database system (2010). R package version 0.1-7. http://www.postgresql.org. Accessed date on 2nd August 2012

40. Cook, R.D., Weisberg, S.: Residuals and Influence in Regression. Chapman and Hall, London (1982)
41. Cornsweet, T.: Visual Perception. Academic, New York (1970)
42. Crozier, W.J.: On the visibility of radiation at the human fovea. J. Gen. Physiol. **34**, 87–136 (1950)
43. Crozier, W.J., Wolf, E.: Theory and measurement of visual mechanisms. IV. On flicker with subdivided fields. J. Gen. Physiol. **27**, 401–432 (1944)
44. Dakin, S.C., Bex, P.J.: Natural image statistics mediate brightness filling in. Proc. Biol. Sci. **1531**, 2341–2348 (2003)
45. Dartnall, H.J.A., Bowmaker, J.K., Mollon, J.D.: Microspectrophotometry of human photoreceptors. In: Mollon, J.D., Sharpe, L.T. (eds.) Colour Vision: Physiology and Psychophysics, pp. 69–80. Academic, London (1983)
46. Davison, A.C., Hinkley, D.V.: Bootstrap Methods and their Applications. Cambridge University Press, Cambridge (1997). http://statwww.epfl.ch/davison/BMA/
47. Debreu, G.: Topological methods in cardinal utility theory. In: Arrow, K.J., Karlin, S., Suppes, P. (eds.) Mathematical Methods in the Social Sciences, pp. 16–26. Stanford University Press, Stanford (1960)
48. DeCarlo, L.T.: Signal detection theory and generalized linear models. Psychol. Meth. **3**, 186–205 (1998)
49. DeCarlo, L.T.: On the statistical and theoretical basis of signal detection theory and extensions: Unequal variance, random coefficient and mixture models. J. Math. Psychol. **54**, 304–313 (2010)
50. Delord, S., Devinck, F., Knoblauch, K.: Surface and edge in visual detection: Is filling-in necessary? J. Vis. **4**, 68a (2004). http://journalofvision.org/4/8/68/
51. Devinck, F.: Les traitements visuels chez l'homme : stratégies de classification de la forme. Ph.D. thesis, Université Lyon 2 (2003)
52. Devroye, L.: Non-uniform Random Variate Generation. Springer, New York (1986)
53. Dobson, A.J.: An Introduction to Generalized Linear Models. Chapman and Hall, London (1990)
54. Edwards, A.W.F.: Likelihood, expanded edn. Johns Hopkins University Press, Baltimore (1992)
55. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman Hall, New York (1993)
56. Emrith, K., Chantler, M.J., Green, P.R., Maloney, L.T., Clarke, A.D.F.: Measuring perceived differences in surface texture due to changes in higher order statistics. J. Opt. Soc. Am. A **27**(5), 1232–1244 (2010)
57. Falmagne, J.C.: Elements of Psychophysical Theory. Oxford University Press, Oxford (1985)
58. Faraway, J.J.: Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models. Chapman and Hall/CRC, Boca Raton (2006)
59. Fechner, G.T.: Elemente der Psychophysik. Druck und Verlag von Breitkopfs, Leipzig (1860)
60. Finney, D.J.: Probit Analysis, 3rd edn. Cambridge University Press, Cambridge (1971)
61. Firth, D.: Bias reduction of maximum likelihood estimates. Biometrika **80**, 27–38 (1993)
62. Fisher, R.A.: The design of experiments. In: Bennett, J.H. (ed.) Statistical Methods, Experimental Design and Scientific Inference, 9th edn. Macmillan, New York (1971)
63. Fleming, R.W., Jäkel, F., Maloney, L.T.: Visual perception of thick transparent materials. Psychol. Sci. **22**, 812–820 (2011)
64. Foster, D.H., Bischof, W.F.: Bootstrap estimates of the statistical accuracy of thresholds obtained from psychometric functions. Spatial Vis. **11**(1), 135–139 (1997)
65. Foster, D.H., Żychaluk, K.: Nonparametric estimates of biological transducer functions. IEEE Signal Process. Mag. **24**, 49–58 (2007)
66. Fründ, I., Haenel, N.V., Wichmann, F.A.: Inference for psychometric functions in the presence of nonstationary behavior. J. Vis. **11**(6), 1–19 (2011)
67. Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F., Bornkamp, B., Hothorn, T.: mvtnorm: Multivariate normal and t distributions (2011). R package version 0.9-9991. http://CRAN.R-project.org/package=mvtnorm. Accessed date on 2nd August 2012

68. Gescheider., G.A.: Psychophysical scaling. Annu. Rev. Psychol. **39**, 169–200 (1988)
69. Glass, L.: Moiré effect from random dots. Nature **223**, 578–580 (1969)
70. GmbH, M.S.: XLConnect: Excel Connector for R (2011). R package version 0.1-5. http://CRAN.R-project.org/package=XLConnect. Accessed date on 2nd August 2012
71. Gold, J.M., Murray, R.F., Bennett, P.J., Sekular, A.B.: Deriving behavioural receptive fields for visually completed contours. Curr. Biol. **10**, 663–666 (2000)
72. Green, D.M., Swets, J.A.: Signal Detection Theory and Psychophysics. Robert E. Krieger Publishing Company, Huntington (1966/1974)
73. Guilford, J.P.: Psychometric Methods, 2nd edn. McGraw-Hill, New York (1954)
74. Hadfield, J.D.: Mcmc methods for multi-response generalized linear mixed models: The MCMCglmm R package. J. Stat. Software **33**(2), 1–22 (2010). http://www.jstatsoft.org/v33/i02/
75. Hansen, T., Gegenfurtner, K.R.: Classification images for chromatic signal detection. J. Opt. Soc. Am. A **22**, 2081–2089 (2005)
76. Harrell, F.E.: rms: Regression modeling strategies (2011). R package version 3.3-1. http://CRAN.R-project.org/package=rms. Accessed date on 2nd August 2012
77. Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman and Hall, London (1990)
78. Hauck, W.W. Jr, Donner, A.: Wald's test as applied to hypotheses in logit analysis. J. Am. Stat. Assoc. **72**, 851–853 (1977)
79. Hecht, S., Shlaer, S., Pirenne, M.H.: Energy, quanta and vision. J. Gen. Physiol. **25**, 819–840 (1942)
80. Ho, Y.X., Landy, M.S., Maloney, L.T.: Conjoint measurement of gloss and surface texture. Psychol. Sci. **19**, 196–204 (2008)
81. James, D.A.: DBI: R Database Interface (2009). R package version 0.2.5. http://CRAN.R-project.org/package=DBI. Accessed date on 2nd August 2012
82. James, D.A.: RSQLite: SQLite interface for R (2011). R package version 0.10.0. http://CRAN.R-project.org/package=RSQLite. Accessed date on 2nd August 2012
83. James, D.A., DebRoy, S.: RMySQL: R interface to the MySQL database (2011). R package version 0.8-0. http://biostat.mc.vanderbilt.edu/RMySQL. Accessed date on 2nd August 2012
84. James, D.A., Luciani, J.: ROracle: Oracle database interface for R (2007). R package version 0.5-9. http://www.omegahat.org. Accessed date on 2nd August 2012
85. Johnson, N.L., Kemp, A.W., Kotz, S.: Univariate Discrete Distributions. Wiley, New York (2005)
86. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, vol. 1. Wiley, New York (1994)
87. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, vol. 2. Wiley, New York (1995)
88. Keppel, G.: Design & Analysis: A Researcher's Handbook, 2nd edn. Prentice-Hall, Englewood Cliffs (1982)
89. Kienzle, W., Franz, M.O., Scholkopf, B., Wichmann, F.A.: Center-surround patterns emerge as optimal predictors for human saccade targets. J. Vis. **9**, 1–15 (2009)
90. Kienzle, W., Wichmann, F.A., Schölkopf, B., Franz, M.O.: A nonparametric approach to bottom-up visual saliency. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, pp. 689–696. MIT, Cambridge (2007)
91. Kingdom, F.A.A., Prins, N.: Psychophysics: A Practical Introduction. Academic, New York (2009)
92. Kleiber, C., Zeileis, A.: Applied Econometrics with R. Springer, New York (2008). http://CRAN.R-project.org/package=AER
93. Klein, S.A.: Measuring, estimating, and understanding the psychometric function: A commentary. Percept. Psychophys. **63**, 1421–1455 (2001)
94. Knoblauch, K.: psyphy: Functions for analyzing psychophysical data in R (2012). R package version 0.1-7. http://cran.r-project.org/web/packages/psyphy

95. Knoblauch, K., Maloney, L.T.: Estimating classification images with generalized linear and additive models. J. Vis. **8**, 1–19 (2008)
96. Knoblauch, K., Maloney, L.T.: MLDS: Maximum likelihood difference scaling in R. J. Stat. Software **25**, 1–26 (2008). http://www.jstatsoft.org/v25/i02
97. Knoblauch, K., Maloney, L.T.: MLCM: Maximum likelihood conjoint measurement (2011). R package version 0.0-8. http://CRAN.R-project.org/package=MLCM. Accessed date on 2nd August 2012
98. Knoblauch, K., Maloney, L.T.: MPDiR: Data sets and scripts for Modeling Psychophysical Data in R (2012). R package version 0.1-11. http://cran.r-project.org/web/packages/MPDiR
99. Knoblauch, K., Vital-Durand, F., Barbur, J.: Variation of chromatic sensitivity across the life span. Vis. Res. **41**, 23–36 (2001)
100. Knuth, D.E.: The Art of Computer Programming: Seminumerical Programming, 3rd edn. Addison-Wesley, New York (1997)
101. Komárek, A., Lesaffre, E.: Generalized linear mixed model with a penalized gaussian mixture as a random-effects distribution. Comput. Stat. Data Anal. **52**(7), 3441–3458 (2008)
102. Kontsevich, L.L., Tyler, C.W.: What makes Mona Lisa smile? Vis. Res. **44**, 1493–1498 (2004)
103. Kosmidis, I.: brglm: Bias reduction in binary-response GLMs (2007). R package version 0.5-6. http://www.ucl.ac.uk/~ucakiko/software.html. Accessed date on 2nd August 2012
104. Krantz, D.H., Luce, R.D., Suppes, P., Tversky, A.: Foundations of Measurement (Vol. 1): Additive and Polynomial Representations. Academic, New York (1971)
105. Kuss, M., Jäkel, F., Wichmann, F.A.: Bayesian inference for psychometric functions. J. Vis. **5**, 478–492 (2005). http://journalofvision.org/5/5/8/
106. Lang, D.T.: Rcompression: In-memory decompression for GNU zip and bzip2 formats. R package version 0.93-2. http://www.omegahat.org/Rcompression. Accessed date on 2nd August 2012
107. Lazar, N.A.: The Statistical Analysis of Functional MRI Data. Springer, New York (2008)
108. Legge, G.E., Gu, Y.C., Luebker, A.: Efficiency of graphical perception. Percept. Psychophys. **46**, 365–374 (1989)
109. Lehmann, E.L., Casella, G.: Theory of Point Estimation, 2nd edn. Springer, New York (1998)
110. Lemon, J: Plotrix: A package in the red light district of R. R-News **6**(4), 8–12 (2010)
111. Levi, D.M., Klein, S.A.: Classification images for detection and position discrimination in the fovea and parafovea. J. Vis. **2**(1), 46–65 (2002)
112. Li, Y., Baron, J.: Behavioral Research Data Analysis in R, 1st edn. Springer, New York (2011)
113. Lindsey, D., Brown, A.: Color naming and the phototoxic effects of sunlight on the eye. Psychol. Sci. **13**, 506–512 (2002)
114. Lindsey, D.T., Brown, A.M., Reijnen, E., Rich, A.N., Kuzmova, Y.I., Wolfe, J.M.: Color channels, not color appearance or color categories, guide visual search for desaturated color targets. Psychol. Sci. **21**, 1208–1214 (2010)
115. Link, S.W.: The Wave Theory of Difference and Similarity. Lawrence Erlbaum Associates, Hillsdale (1992)
116. Luce, R.D., Green, D.M.: Parallel psychometric functions from a set of independent detectors. Psychol. Rev. **82**, 483–486 (1975)
117. Luce, R.D., Tukey, J.W.: Simultaneous conjoint measurement: A new scale type of fundamental measurement. J. Math. Psychol. **32**, 466–473 (1964)
118. Macke, J.H., Wichmann, F.A.: Estimating predictive stimulus features from psychophysical data: The decision image technique applied to human faces. J. Vis. **10** (2010)
119. MacLeod, D.: Visual sensitivity. Annu. Rev. Psychol. **29**, 613–645 (1978)
120. Macmillan, N.A., Creelman, C.D.: Detection Theory: A User's Guide, 2nd edn. Lawrence Erlbaum Associates, New York (2005)
121. Maloney, L.T.: Confidence intervals for the parameters of psychometric functions. Percept. Psychophys. **47**(2), 127–134 (1990)
122. Maloney, L.T., Dal Martello, M.F.: Kin recognition and the perceived facial similarity of children. J. Vis. **6**, 1047–1056 (2006). http://journalofvision.org/6/10/4/

123. Maloney, L.T., Yang, J.N.: Maximum Likelihood difference scaling. J. Vis. **3**(8), 573–585 (2003). http://www.journalofvision.org/3/8/5
124. Mamassian, P., Goutcher, R.: Temporal dynamics in bistable perception. J. Vis. **5**, 361–375 (2005)
125. Mangini, M.C., Biederman, I.: Making the ineffable explicit: Estimating the information employed for face classifications. Cognit. Sci. **28**, 209–226 (2004)
126. Marin-Franch, I., Żychaluk, K., Foster, D.H.: modelfree: Model-free estimation of a psychometric function (2010). R package version 1.0. http://CRAN.R-project.org/package=modelfree. Accessed date on 2nd August 2012
127. McCullagh, P., Nelder, J.A.: Generalized Linear Models. Chapman and Hall, London (1989)
128. Mineault, P.J., Barthelme, S., Pack, C.C.: Improved classification images with sparse priors in a smooth basis. J. Vis. **9**, 1–24 (2009)
129. Mood, A., Graybill, F.A., Boes., D.C.: Introduction to the Theory of Statistics, 3rd edn. McGraw-Hill, New York (1974)
130. Murray, R.F.: Classification images: A review. J. Vis. **11**, 1–25 (2011)
131. Murray, R.F., Bennett, P.J., Sekuler, A.B.: Optimal methods for calculating classification images: Weighted sums. J. Vis. **2**(1), 79–104 (2002)
132. Murrel, P.: Introduction to Data Technologies. Chapman and Hall/CRC, Boca Raton (2009)
133. Nandy, A.S., Tjan, B.S.: The nature of letter crowding as revealed by first- and second-order classification images. J. Vis. **7**(2), 5.1–26 (2007)
134. Neri, P.: Estimation of nonlinear psychophysical kernels. J. Vis. **4**, 82–91 (2004). http://journalofvision.org/4/2/2/
135. Neri, P.: How inherently noisy is human sensory processing? Psychonomic Bull. Rev. **17**, 802–808 (2010)
136. Neri, P., Heeger, D.J.: Spatiotemporal mechanisms for detecting and identifying image features in human vision. Nat. Neurosci. **5**, 812–816 (2002)
137. Neri, P., Parker, A.J., Blakemore, C.: Probing the human stereoscopic system with reverse correlation. Nature **401**, 695–698 (1999)
138. Newsome, W.T., Britten, K.H., Movshon, J.A.: Neuronal correlates of a perceptual decision. Nature **341**(6237), 52–54 (1989)
139. Obein, G., Knoblauch, K., Viénot, F.: Difference scaling of gloss: Nonlinearity, binocularity, and constancy. J. Vis. **4**(9), 711–720 (2004)
140. Peirce, J.W.: The potential importance of saturating and supersaturating contrast response functions in visual cortex. J. Vis. **7**, 13 (2007)
141. Pinheiro, J., Bates, D.: Mixed-Effects Models in S and S-PLUS. Springer, New York (2000)
142. Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., the R Development Core Team: nlme: Linear and nonlinear mixed effects models (2012). R package version 3.1-104
143. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes, 3rd ed.: The Art of Scientific Computing. Cambridge University Press, Cambridge (2007)
144. Quick, R.: A vector-magnitude model of contrast detection. Kybernetik **16**, 65–67 (1974)
145. R-core members, et al.: Foreign: Read data stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...(2011). R package version 0.8-46. http://CRAN.R-project.org/package=foreign. Accessed date on 2nd August 2012
146. R Development Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2011). ISBN 3-900051-07-0. http://www.R-project.org/. Accessed date on 2nd August 2012
147. Rensink, R.A., Baldridge, G.: The perception of correlation in scatterplots. Comput. Graph. Forum **29**, 1203–1210 (2010)
148. Rhodes, G., Maloney, L.T., Turner, J., Ewing, L.: Adaptive face coding and discrimination around the average face. Vis. Res. **47**, 974–989 (2007)
149. Ripley, B.: RODBC: ODBC Database Access (2011). R package version 1.3-3. http://CRAN.R-project.org/package=RODBC. Accessed date on 2nd August 2012
150. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (1996)

151. Roberts, F.S.: Measurement Theory. Cambridge University Press, Cambridge (1985)
152. Ross, M.G., Cohen, A.L.: Using graphical models to infer multiple visual classification features. J. Vis. **9**(3), 23.1–24 (2009)
153. Ross, S.M.: A First Course in Probability Theory, 8th edn. Prentice-Hall, Englewood Cliffs (2009)
154. Rouder, J.N., Lu, J., Sun, D., Speckman, P., Morey, R., Naveh-Benjamin, M.: Signal detection models with random participant and item effects. Psychometrika **72**, 621–642 (2007)
155. Salsberg, D.: The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century. W. H. Freeman, New York (2001)
156. Sarkar, D.: Lattice: Multivariate Data Visualization with R. Springer, New York (2008). http://lmdvr.r-forge.r-project.org
157. Schneider, B.: Individual loudness functions determined from direct comparisons of loudness intervals. Percept. Psychophys. **28**, 493–503 (1980)
158. Schneider, B.: A technique for the nonmetric analysis of paired comparisons of psychological intervals. Psychometrika **45**, 357–372 (1980)
159. Schneider, B., Parker, S., Stein, D.: The measurement of loudness using direct comparisons of sensory intervals. J. Math. Psychol. **11**, 259–273 (1974)
160. Schwartz, M.: WriteXLS: Cross-platform Perl based R function to create Excel 2003 (XLS) files (2010). R package version 2.1.0. http://CRAN.R-project.org/package=WriteXLS. Accessed date on 2nd August 2012
161. Solomon, J.A.: Noise reveals visual mechanisms of detection and discrimination. J. Vis. **2**(1), 105–120 (2002)
162. Spector, P.: Data Manipulation with R. Springer, New York (2008)
163. van Steen, G.: dataframes2xls: dataframes2xls writes data frames to xls files (2011). R package version 0.4.5. http://cran.r-project.org/web/packages/dataframes2xls. Accessed date on 2nd August 2012
164. Stevens, S.S.: On the theory of scales of measurement. Science **103**, 677–680 (1946)
165. Stevens, S.S.: On the psychophysical law. Psychol. Rev. **64**, 153–181 (1957)
166. Strasburger, H.: Converting between measures of slope of the psychometric function. Percept. Psychophys. **63**, 1348–1355 (2001)
167. Sun, H., Lee, B., Baraas, R.: Systematic misestimation in a vernier task arising from contrast mismatch. Vis. Neurosci. **25**, 365–370 (2008)
168. Suppes, P.: Finite equal-interval measurement structures. Theoria **38**, 45–63 (1972)
169. Tanner, W.P., Swets, J.A.: A decision-making theory of visual detection. Psychol. Rev. **61**, 401–409 (1954)
170. Teller, D.Y.: The forced-choice preferential looking procedure: A psychophysical technique for use with human infants. Infant Behav. Dev. **2**, 135–153 (1979)
171. Thibault, D., Brosseau-Lachaine, O., Faubert, J., Vital-Durand, F.: Maturation of the sensitivity for luminance and contrast modulated patterns during development of normal and pathological human children. Vis. Res. **47**, 1561–1569 (2007)
172. Thomas, J.P., Knoblauch, K.: Frequency and phase contributions to the detection of temporal luminance modulation. J. Opt. Soc. Am. A **22**(10), 2257–2261 (2005)
173. Thurstone, L.L.: A law of comparative judgement. Psychol. Rev. **34**, 273–286 (1927)
174. Tolhurst, D.J., Movshon, J.A., Dean, A.F.: The statistical reliability of signals in single neurons in cat and monkey visual cortex. Vis. Res. **23**(8), 775–785 (1983)
175. Treutwein, B., Strasburger, H.: Fitting the psychometric function. Percept. Psychophys. **61**(1), 87–106 (1999)
176. Urban, F.M.: The method of constant stimuli and its generalizations. Psychol. Rev. **17**, 229–259 (1910)
177. Urbanek, S.: RJDBC: Provides access to databases through the JDBC interface (2011). R package version 0.2-0. http://www.rforge.net/RJDBC/. Accessed date on 2nd August 2012
178. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S, 4th edn. Springer, New York (2002). http://www.stats.ox.ac.uk/pub/MASS4

179. Victor, J.: Analyzing receptive fields, classification images and functional images: Challenges with opportunities for synergy. Nat. Neurosci. **8**, 1651–1656 (2005)
180. Warnes, G.R.: gmodels: Various R programming tools for model fitting (2011). R package version 2.15.1. http://CRAN.R-project.org/package=gmodels. Accessed date on 2nd August 2012
181. Warnes, G.R., et al.: gdata: Various R programming tools for data manipulation (2010). R package version 2.8.1. http://CRAN.R-project.org/package=gdata. Accessed date on 2nd August 2012
182. Watson, A.: The spatial standard observer: A human vision model for display inspection. In: 353 SID Symposium Digest of Technical Papers, **37**, pp. 1312–1315 (2006)
183. Watson, A.B., Ahumada, A.J.: A standard model for foveal detection of spatial contrast. J. Vis. **5**, 717–740 (2005)
184. Watson, A.B., Pelli, D.G.: QUEST: A Bayesian adaptive psychometric method. Percept. Psychophys. **33**, 113–120 (1983)
185. Watson, G.A.: Approximation Theory and Numerical Methods. Wiley, New York (1980)
186. Westheimer, G.: The spatial grain of the perifoveal visual field. Vis. Res. **22**(1), 157–162 (1982)
187. Wichmann, F.A., Graf, A.B.A., Simoncelli, E.P., Bülthoff, H.H., Schölkopf, B.: Machine learning applied to perception: Decision-images for gender classification. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 17, pp. 1489–1496. MIT, Cambridge (2005)
188. Wichmann, F.A., Hill, N.J.: The psychometric function: I. fitting, sampling and goodness of fit. Percept. Psychophys. **63**, 1293–1313 (2001)
189. Wickens, T.D.: Elementary Signal Detection Theory. Oxford University Press, New York (2002)
190. Wickham, H.: ggplot2: Elegant Graphics for Data Analysis. Springer, New York (2009). http://had.co.nz/ggplot2/book
191. Wilkinson, G.N., Rogers, C.E.: Symbolic description of factorial models for analysis of variance. Appl. Stat. **22**, 392–399 (1973)
192. Wilkinson, L.: The Grammar of Graphics. Springer, New York (2005)
193. Wilks, S.S.: The large-sample distribution of the likelihood ratio for testing composite hypotheses. Ann. Math. Stat. **9**, 60–62 (1938)
194. Williams, J., Ramaswamy, D., Oulhaj, A.: 10 Hz flicker improves recognition memory in older people. BMC Neurosci. **7**, 21 (2006)
195. Winer, B.J.: Statistical Principles in Experimental Design, 2nd edn. McGraw-Hill, New York (1971)
196. Winship, C., Mare, R.D.: Regression models with ordinal variables. Am. Soc. Rev. **49**, 512–525 (1984)
197. Wood, S.: gamm4: Generalized additive mixed models using mgcv and lme4 (2011). R package version 0.1-3. http://CRAN.R-project.org/package=gamm4. Accessed date on 2nd August 2012
198. Wood, S.N.: Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC, Boca Raton (2006)
199. Xie, Y., Griffin, L.D.: A 'portholes' experiment for probing perception of small patches of natural images. Perception **36**, 315 (2007)
200. Yang, J.N., Szeverenyi, N.M., Ts'o, D.: Neural resources associated with perceptual judgment across sensory modalities. Cerebr. Cortex **18**, 38–45 (2008)
201. Yee, T.W.: The vgam package for categorical data analysis. J. Stat. Software **32**(10), 1–34 (2010). http://www.jstatsoft.org/v32/i10
202. Yeshurun, Y., Carrasco, M., Maloney, L.T.: Bias and sensitivity in two-interval forced choice procedures: Tests of the difference model. Vis. Res. **48**, 1837–1851 (2008)
203. Yovel, Y., Franz, M.O., Stilz, P., Schnitzler, H.U.: Plant classification from bat-like echolocation signals. PLoS Comput. Biol. **4**, e1000032 (2008)

204. Yssaad-Fesselier, R., Knoblauch, K.: Modeling psychometric functions in R. Behav. Res. Meth. Instrum. Comp. **38**, 28–41 (2006)
205. Zhaoping, L., Jingling, L.: Filling-in and suppression of visual perception from context: A Bayesian account of perceptual biases by contextual influences. PLoS Comput. Biol. **4**, e14 (2008)
206. Zucchini, W.: An introduction to model selection. J. Math. Psychol. **44**, 41–61 (2000)
207. Zuur, A., Ieno, E.N., Walker, N., Saveiliev, A.A., Smith, G.M.: Mixed Effects Lodels and Extensions in Ecology with R. Springer, New York (2009)
208. Żychaluk, K., Foster, D.H.: Model-free estimation of the psychometric function. Attention Percept. Psychophys. **71**, 1414–1425 (2009)

Index

K. Knoblauch and L.T. Maloney, *Modeling Psychophysical Data in R*, Use R! 32,
DOI 10.1007/978-1-4614-4475-6,