Part I

The Basics

Part I is exactly what its title suggests – a basic introduction to the analysis of time series data. We begin in Chapter 1 by providing an intuitive definition of a time series data set, and with a discussion on the main objectives of time series analysis. In Chapter 2 we present basic descriptive statistics that are used for describing the behavior of a second-order stationary series. Chapter 3 concludes Part I by a presentation of common transformation and nonparametric prediction methods for time series.

It should not go without stating that, throughout Part I we assume the time series is second-order stationary, without actually stating that assumption has been made, or trying to define that concept. Because we do not want to provide the illusion that these methods are appropriate in all cases, we allude to the fact that there is more to consider throughout the chapters. However, an in depth discussion of stationarity and time series probability, along with time series models, and statistical inference for time series will be address in Part II.

Chapter 1

Introduction to Time Series

Almost every area of scientific inquiry is concerned with data collected over time; that is, with time series. Figure 1.1 contains the graph of ten such time series. A listing of all the time series analyzed in this book is included in Appendix C, and the web site that accompanies the book includes a file for each data set.

These series were chosen as a beginning point for the discussion in the book because they illustrate some of the different type of series with which we will be concerned and the scientific questions that one can answer using time series analysis. The aim of this chapter is to acquaint the reader with some of the basic ideas of time series analysis and to introduce the *Timeslab in R* software that accompanies this book.

1.1 Time Series Data Types

Time series can be classified in many ways, including by the following four characteristics:

1. **Dimension of the index set.** Associated with all time series is an index set we will denote by $\mathcal{T}$. The set $\mathcal{T}$ can be one-dimensional or multidimensional. Daily births of females in California (Series I) have a one-dimensional index set. In contrast, wheat yields recorded over a regular grid of positions in a large field have a two-dimensional index set. This second example also illustrates that the index set need not literally be "time," but can also be "location." When the index set is two-dimensional, the observed data is more appropriately analyzed using methods in spatial statistics, a subject area not addressed in this text. Spatio-temporal processes have at least a three-dimensional index set, indicating the time of the observation, as well as the position.

2. **Nature of the index set.** The index set $\mathcal{T}$ can be continuous or discrete. A digital electroencephalogram (EEG) that displays measurements of the electrical activity in the brain is a continuous one-dimensional series. The daily birth data is an example of a discrete one-dimensional time series. Discrete-time time series can have observations

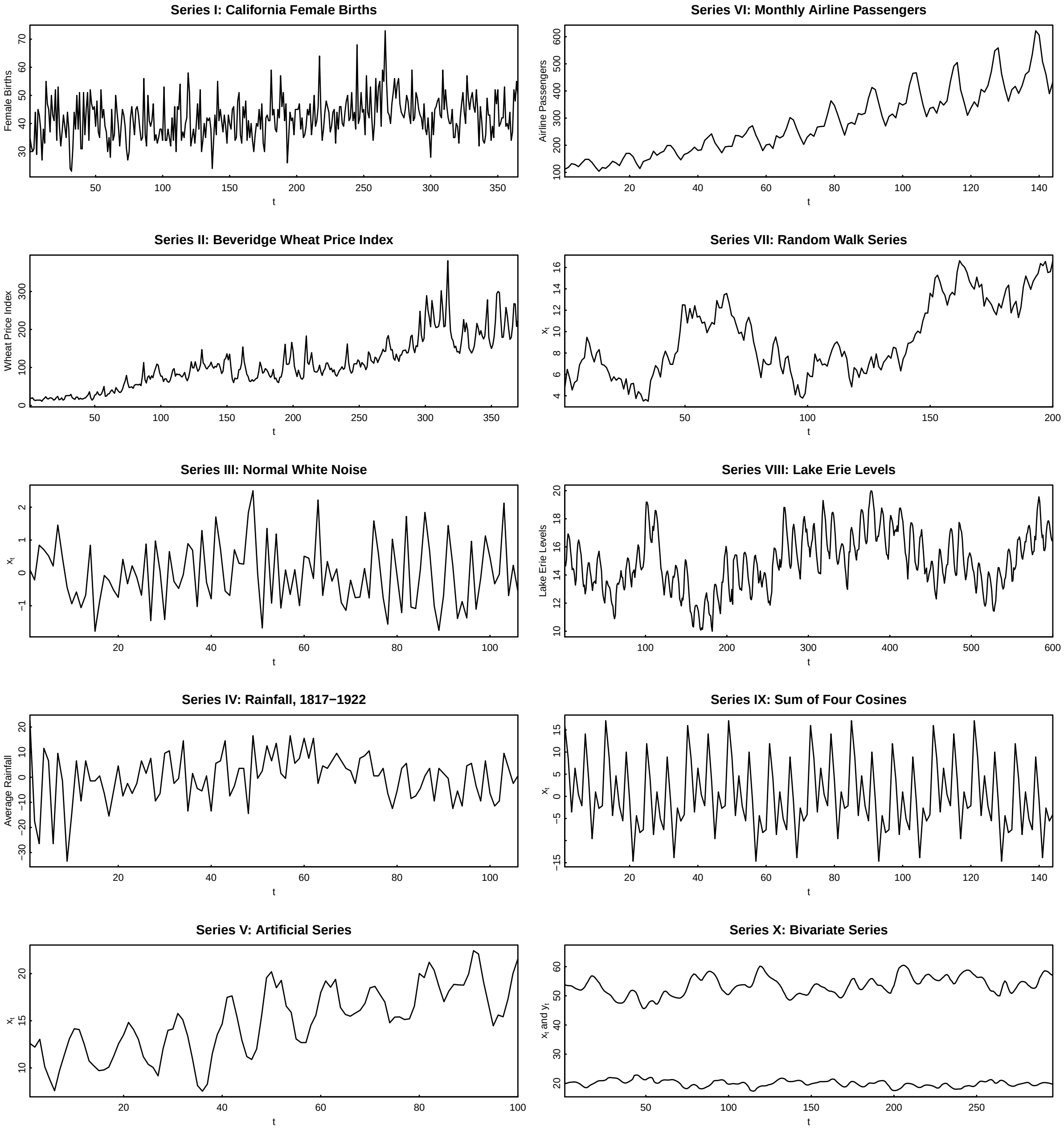

Figure 1.1: Some examples of time series.

that are equally spaced or irregularly spaced in time. In Section 2.1, we explain a distinction in time series notation that is based on the nature of the index set.

3. **Univariate (single variable, scalar) or multivariate (many variables).** Univariate (also called scalar) time series have one variable at each time. On the other hand, multivariate (or vector) time series have a vector of variables measured at each time. For example, a series consisting of monthly interest rates and the gross national product is a one-dimensional bivariate time series. Series X is another such example, as the input and output values of a gas in a chemical process are recorded at each time.

4. **Discrete or continuous variable space.** Whether the time series variable at each time is discrete or continuous is the fourth means of classification. Most series are continuous; that is, each variable can take on any in an interval of real numbers. An example of a discrete-valued series is the series of monthly airline passengers (Series VI), or a binary time series, in which the series can take on only one of two possible values.

In this book, we will be concerned almost exclusively with time series that are one dimensional, equally spaced in discrete time, with a continuous variable space, and are either univariate (Chapters 1-3) or bivariate (Chapter 4).

1.2 Time Series Memory Types

The basic property of time series analysis is that it is concerned with repeated measurements on the same phenomenon at different times. Because of this, the analyst must take into account the correlation between successive observations. This is in direct contrast to the data analyzed in elementary statistics courses where the data are assumed to be independent and identically distributed, and are obtained randomly by sampling some population or populations. The presence of correlation makes the analysis of time series data and the interpretation of the results much more difficult than in the independent case.

In this section, we will classify time series into three broad classes based on what we will call their "memory type."

1. **Purely random series.** This type of series shows no patterns over time. Series III is an example of such a series. It was created using a function in R that generates random numbers. This particular series is indistinguishable from a random sample from a standard normal distribution. In contrast to this, Series IV is a real data set of monthly total rainfall. Series IV appears quite similar to Series III. We will see later that series like III and IV are aptly named "white noise." Purely random series are also called **no memory series** since one characterization of the statistical independence is that an observation at one time has no memory of the observation at any other time.

2. **Long-memory series.** This type of series is the opposite extreme of white noise; that is, a plot of the data looks to be almost that of a deterministic function of time.

Series V and VI illustrate this type. The first was artificially generated as values lying on a cosine curve that goes through ten cycles with small random numbers added to each point. Series VI is a real economic time series (monthly total international airline passengers for 12 years). These two series have in common that both could be almost perfectly extrapolated far into the future unless something were to happen to the mechanism generating the data. This is the origin of the term "long memory." The dependence on the past does not die away quickly. note that many of the time series in business and economics are long memory. Long memory series are also referred to as series with **long-range dependence**.

3. **Short-memory series.** This type lies between white noise and long memory, occurs most often in the physical and engineering sciences, and comprises the bulk of time series that can be most usefully analyzed by the sophisticated methods of time series analysis that we will study. Series VIII and IX appear to be short-memory series; clearly observations close together in time are more similar than those far apart in time, but there are no apparent deterministic patterns in the data (although upon closer inspection, you might be able to tell that Series IX is actually the sum of four cosine curves having squared amplitudes 10, 13, 37, and 65 and going through 4, 12, 24, and 48 cycles, respectively). In a short-memory series the predictability of the observations at one place in time from past observations appears to die out quickly as time goes on.

Time series can also be classified as **stationary** or **non-stationary**. We will study these concepts in detail in Chapter 4, but for now, we will think of nonstationary as long-memory and stationary as white noise or short memory.

1.3 Aims of Time Series Analysis

The aims of time series analysis can be roughly grouped into four categories: descriptive, inferential, predictive, and control. Descriptive time series analysis is extremely graphical in nature because of the lack of independence among the quantities that it investigates. In Chapter 2, we will consider three fundamental graphs that can be used as the primary steps in classifying a time series into long, short, or no memory.

Chapter 3 addresses two related topics: time series transformations and basic forecasting methods. In that chapter, we will discuss reasons, methods, and results of transforming a series. Many transformations yield themselves to simple extension that will provide forecasts of the series. So these basic forecasting methods are also presented in Chapter 3.

In Chapter 6, we will consider the inferential part of univariate time series analysis. Essentially we seek to determine from one time series data set what other data could have been observed or are expected to be observed in the future. To do this, we must be able to assume that the random mechanism generating the observed data follows some kind of model. Basic time series models are introduced in Chapter 4.

The third aim is to use the correlation in the time series to predict the future. In Chapter 4 we introduce the basic theory of prediction, and in Chapter 6, we give several examples of model-based prediction methods. Given the ability to predict the future of a time series, one naturally begins to think of controlling the future. Thus one can begin adjusting the values of time series so that the future values of the series are in line with what is desired.

1.4 Using *Timeslab in R* and *R*

Close to the end of most chapters in this book is a section that documents the different functions that will help you to study the concepts in that chapter. The section will be broken down into two subsections. The first will explain functions that are a part of *Timeslab in R*, and the second section will describe functions that are a part of the standard release of *R* that are most helpful to time series analysis. If you're not familiar with the *R* language, you are encouraged to read the tutorial on using *R* found in Appendix D.

1.4.1 *Timeslab in R* Functions

Timeslab in R is an updated version of the first release of *Timeslab* (Newton 1988), which was originally written predominantly in a programming language known as FORTRAN 77. That software has been updated to interface with *R* and accordingly, has received new name, *Timeslab in R*. Succinctly, *Timeslab in R* is a set of functions written in the *R* language and in FORTRAN 95. (Don't worry. You don't have to know anything about FORTRAN to use *Timeslab in R*.) The functions are specifically designed for time series analysis.

The `datasets` Function

The `datasets` function is a *Timeslab* function that, when called, returns all the datasets considered in this text in a list. The function has no arguments. The call to the function is

```
datasets()
```

The different datasets, which comprise the items in the list are briefly discussed below. More thorough descriptions of the datasets are provided throughout the text and in Appendix C. Each of the time series below is stored in a vector.

`air`: monthly number of airline passengers from 1949 to 1960, $n = 144$.

`artif`: an artificial data set of length $n = 100$. A description of how this data set was formed is found in Section 3.1.2.

`bev`: Beveridge wheat price index from 1500 to 1869, compiled by Beveridge (1921), $n = 370$.

`buffsnow`: yearly snowfall in Buffalo, New York from 1910 to 1972, $n = 63$.

`calfem`: daily number of females born in California in 1959, $n = 365$.

`cos4`: sum of four cosines, $n = 144$. A more thorough description of how this data set was formed is found in Section 2.2.3.

`cradfq`: median noon hour value of what are called critical radio frequencies each month in Washington, D.C., for the period May 1934 to April 1954. These frequencies reflect the highest radio frequency that can be used for broadcasting. See Siddiqui (1962) for further discussion of these data. This series is often considered as the first component in the bivariate series having second component of total monthly number of sunspots for the corresponding time period (`cradfqsun`). The length of the series is $n = 240$.

`cradfqsun`: monthly number of sunspots from May 1934 to April 1954. This series is often considered as the second component in the bivariate series having first component as the critical radio frequencies each month in Washington, D.C. for the corresponding period (`cradfq`). The length of the series is $n = 240$.

`eriel`: monthly level of Lake Erie from 1921 to 1970, $n = 600$.

`gasfurnin`: Methane gas feed rate into a gas furnace, $n = 296$. This is the first component in a bivariate series having second component being carbon monoxide concentration output from the same furnace.

`gasfurnout`: Carbon monoxide concentration output from a gas furnace with a methane gas input feed, $n = 296$. The methane gas input feed is the first component of this bivariate series.

`lh1`: The first component in a bivariate series of measurements of luteinizing hormone (LH) at 10-minute intervals for two difference cows each at day three of their estrous cycle.

`lh2`: The second component in a bivariate series of measurements of luteinizing hormone (LH) at 10-minute intervals for two difference cows each at day three of their estrous cycle.

`lynx`: the number of Canadian lynx furs caught by the Hudson Bay company between 1821 and 1935 as reported by Elton & Nicholson (1942).

`mlco2`: monthly measurements of carbon dioxide above Mauna Loa, Hawaii, $n = 216$.

`normwn`: normal white noise series, $n = 106$.

`nyct`: The first component of a bivariate series consisting of the average temperatures in New York City for $n = 168$ months from 1949 to 1959. The second component is the monthly number of babies delivered in New York City during the same time period.

`nycb`: The second component in a bivariate series consisting of the monthly number of babies delivered in New York City from 1949 to 1959. The first component is the average monthly temperature in New York City for the same time period.

`raineast`: average rainfall in the eastern United States from 1817 to 1922, $n = 106$.

`rw`: random walk of length $n = 200$ formed from the cumulative sums of a normal white noise series.

`sales`: monthly sales of a company from January, 1965 to May, 1971, $n = 77$.

`star`: brightness of a star on $n = 600$ successive midnights.

`waves`: forces on a cylinder suspended in a tank of water every 0.15 seconds, $n = 320$.

`wolfer`: Wolfer's annual sunspot numbers from 1749 to 1963, $n = 215$

Example 1.1 USING THE `datasets` FUNCTION.
The result of issuing the R *command*

```
x.air <- datasets()$air
```

is to call the function `datasets`*, and to take only the item called* `air` *in the returned list, and assign it to a new variable* `x.air`*.*

The `wn` Function

The *Timeslab in R* function

```
x <- wn(n, dist = 1, seed = 0)
```

will generate a sample realization of length `n` from a white noise series (using the random number generator seed entered in the scalar argument `seed`) for any of the distributions in Table 1.4.1. If the value of `seed = 0` (the default), then a seed is randomly generated. The default distribution is the standard normal distribution, for which the value of `dist = 1`, or `dist = "normal"`. If the character-value is supplied for the `dist` argument, it may be truncated to the first four unique characters (as they appear in the table). Six of the eight `wn` distributions have an *R* analog. The two that do not are the extreme value and double exponential distributions. For the normal distribution, the *R* function is `rnorm(n)`, for uniform `runif(n)`, for exponential `rexp(n)`, for logistic r̃logis(n), for Cauchy `rcauchy(n)`, and for lognormal white noise, the *R* function call is `lnorm(n)`.

1.4.2 *R* Functions

In this subsection, we will describe only the `plot` function, which is a part of the standard release of *R*. There are many nuances to using and customizing the appearances of graphing windows, and the plots within, that we do not describe here. For more information on those, the interested reader is refered to the help that is a part of *R*, and specifically help for the functions `par`, `axis`, `mtext`, and `plotmath`.

Table 1.1: Distributions of the `wn` function.

`dist`	Distribution and Transformation	Mean	Variance
1, `norm`, or `normal`	N(0,1): Z	0	1
2, `unif`, `uniform`	U(0,1): U	1/2	1/12
3, `expo`, or `exponential`	Exponential: $-\log(1-U)$	1	1
4, `logi`, `logistic`	Logistic: $\log(U/(1-U))$	0	$\pi^2/3$
5, `cauc`, `cauchy`	Cauchy: $\tan(\pi(U-0.5))$	∞	∞
6, `extv` `extvalue`	Extreme value: $\log(-\log(1-U))$ or Gumbel	-0.5772	$\pi^2/6$
7, `logn` or `lognormal`	Lognormal: e^Z	$e^{1/2}$	$e(e-1)$
8, `dexp` or `dexponential`	Double exponential: $\begin{cases} \log(2U) & U \leq 0.5 \\ -\log(2(1-U)) & U > 0.5 \end{cases}$ or LaPlace		

The `plot` Function

The `plot` function is a built-in *R* function that is not a part of the *Timeslab in R* package. However, since plotting the descriptive statistics is so fundamental to time series analysis, the `plot` function is also fundamentally important. The syntax of the plot function is below.

```
plot(x, y, ...)
```

where

`x`: is a set of coordinates of points in the plot. If `x` is the only specified set of coordinates, then `x` is taken to be the set of vertical coordinates. If a second set of coordinates is supplied, then `x` is taken to be the set of horizontal coordinates.

`y`: is the set of y (vertical) coordinates of points in the plot.

`...`: arguments to be passed to methods, such as graphical parameters (see the *R* function `par`). A partial listing of the type of arguments most methods will accept are the following.

`type`: specifies what type of plot should be drawn. Some of the possible types are

`p`: for points (the default),

`l`: for lines,

`b`: for both points and lines,

`n`: for no plotting symbol.

`main`: an overall title for the plot.

`sub`: a sub-title for the plot.

`xlab`: a title for the x-axis.

`ylab`: a title for the y-axis.

`xlim`: a vector of length two with elements specifying the minimum and maximum values for the scale of the x-axis, respectively.

`ylim`: a vector of length two with elements specifying the minimum and maximum values for the scale of the y-axis, respectively.

Example 1.2 Plotting the Airline Data.
Refer back to Example 1.1. The variable created in that example `x.air` *contains the airline data set described in Section 1.4.1. To plot that data set, issue the command below.*

```
plot(x.air, type = "l", main = "Time Plot of Airline Data",
     xlab = "Time", ylab = "Monthly Number of Airline Passengers")
```

Example 1.3 Generating and Plotting White Noise.
There are two parts to this example. The final plot will be the same for both graphs. Issue the following two commands from the R *command prompt.*

```
par(mfrow = c(2, 1))
x <- wn(250, seed = 12345)
plot(x, type = "l", xlab = "t", ylab = expression(paste(x[t])),
     main = "Normal White Noise")
```

The line `par(mfrow = c(2, 1))` *sets a plotting* `par`*ameter of* `multiple figure rows`*. In essence, this divides the plotting window into* 2 *rows (and* 1 *column). Each divisions will contain a new plot. You will see when you issue the* `plot` *function that the resulting graph is in the top half of the plotting window. Now issue the command*

```
plot(wn(250, seed = 12345), type = "l", main = "Normal White Noise",
     xlab = "t", ylab = expression(paste(x[t])))
```

This plot appears in the bottom half of the plotting window. The appearance of this plot is identical to the appearance of the plot in the top half of the window. The data is the same because the value of `seed = 12345` *in both calls to the* `wn` *function. If* `seed = 0`*, the data generated from the two calls would have been different. In either case, the data is generated from a standard normal distribution.*

1.5 Exercises

1.5.1 Theoretical Exercises

The results in these theoretical exercises will be helpful in completing proofs in theoretical exercises in some of the following chapters.

T1.1 Prove the following by induction.

(a) $\sum_{t=1}^{n} t = \dfrac{n(n+1)}{2}$

(b) $\sum_{t=1}^{n} t^2 = \dfrac{n(n+1)(2n+1)}{6}$

(c) $\sum_{t=1}^{n} t^3 = \dfrac{n^2(n+1)^2}{4}$

(d) $\sum_{t=1}^{n} t^4 = \dfrac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

T1.2 Let $f(\theta) = a\cos\theta + b\sin\theta$, and let $c = \sqrt{a^2+b^2}$. Define the arctan2 function of a and b by

$$\phi = \operatorname{arctan2}(b, a) = \begin{cases} \arctan(b/a) & \text{if } a > 0 \\ \arctan(b/a) + \operatorname{sgn}(b)\pi & \text{if } a < 0 \\ \operatorname{sgn}(b)\dfrac{\pi}{2} & \text{if } a = 0, \end{cases} \tag{1.1}$$

where $\operatorname{sgn}(b)$ is 1 for $b \geq 0$ and is -1 for $b < 0$, while arctan is the usual inverse tangent function which takes on values only in the interval $(-\pi/2, \pi/2)$. (For more specific information on the arctangent function, see equation (A.6) of Appendix A.1.) Show that for all a and b,

$$a = c\cos\phi, \quad b = c\sin\phi,$$

and thus

$$f(\theta) = c\cos(\theta - \phi).$$

(*Hint:* $\cos(\theta_1 - \theta_2) = \cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2$.) *Throughout this book, when we write arctan, we mean arctan2.*

T1.3 Show that the sinusoid

$$f(x) = a\cos\frac{2\pi x}{p} + b\sin\frac{2\pi x}{p}$$

has maximum values at $x = (p\phi/(2\pi)) + kp$, for any integer k, where $\phi = \operatorname{arctan2}(b/a)$. Show that if $a > 0$, then $\phi/(2\pi)$ (or $1 + \phi/(2\pi)$ if $b < 0$) is the fraction of a cycle that f is shifted to the right of $a\cos 2\pi x/p$. Thus show that if $a > 0$, then f can only be shifted to the right of $a\cos 2\pi x/p$ a fraction of a cycle contained in either the interval $(0, 1/4)$ or the interval $(3/4, 1)$. Show the same thing is true for $a < 0$.

T1.4 Show that if a and b are independent random variables each having the N(0,1) distribution, then $c^2 = a^2 + b^2$ and $\theta = \arctan(b/a)$ are independent random variables with c^2 being χ^2_2 and θ being $U(-\pi/2, \pi/2)$.

T1.5 Show that $e^{i\theta} = \cos\theta + i\sin\theta$, and thus that $e^{2\pi ik} = 1$ for any integer k.

T1.6 Show that the zeros of the polynomials

$$g(z) = \sum_{j=0}^{p} \alpha_j z^j \quad \text{and} \quad h(z) = \sum_{j=0}^{p} \alpha_j z^{p-j}$$

are reciprocals of each other.

T1.7 If we form m independent confidence intervals, each having confidence coefficient $1-\alpha$, what is the probability that all m of the intervals will include the value of the parameter that they are the confidence interval for? Thus, what should α be in order to have 20 confidence intervals that have joint probability 0.95 of including the true value of their parameter?

1.5.2 Computational Exercises

C1.1 Use the `wn` function to generate four white noise time series from a standard normal distribution. The lengths of the respective series should be `n = 50, 100, 250`, and `500`. Plot the four series in a 2×2 plot array in the same window. Describe the appearance of each graph, and compare the appearance of the four series. In your comparison, be sure to comment on features like the range of the data (displayed on the vertical axis), the center of the observations, and how the changing sample size affects the appearance of the graph.

C1.2 Repeat Exercise C1.1, but instead let the distribution of the data be the uniform(0,1) distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercise C1.1.

C1.3 Repeat Exercise C1.1, but instead let the distribution of the data be the exponential(1) distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 and C1.2.

C1.4 Repeat Exercise C1.1, but instead let the distribution of the data be the logistic(0,1) distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 through C1.3.

C1.5 Repeat Exercise C1.1, but instead let the distribution of the data be the Cauchy distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 through C1.4.

C1.6 Repeat Exercise C1.1, but instead let the distribution of the data be the extreme value distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 through C1.5.

C1.7 Repeat Exercise C1.1, but instead let the distribution of the data be the lognormal distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 through C1.6.

C1.8 Repeat Exercise C1.1, but instead let the distribution of the data be the double exponential distribution. Additionally compare the appearance of the graphs in this exercise with those in Exercises C1.1 through C1.7.

C1.9 Write an *R* function to accomplish the following tasks.

(a) Create a sequence `theta` of 501 evenly spaced values between 0 and 2π.

(b) Create in the same window the six plots listed below. For every plot, use `ylim = c(-2,2)`.

(1) Plot 1: cos `theta` versus `theta`

(2) Plot 2: cos 2`theta` versus `theta`

(3) Plot 3: cos `theta`/2 versus `theta`

(4) Plot 4: 2cos `theta` versus `theta`

(5) Plot 5: 2cos 2`theta` versus `theta`

(6) Plot 6: 2cos `theta`/2 versus `theta`

What is the effect of multiplying `theta` by a constant? What is the effect of multiplying the cosine function by a constant?

C1.10 Use the *R* `plot` function, along with the `lines` function to superimpose plots of a cosine and a sine, each of length 100 with amplitude 1 and period 20. How much out of phase are the two plots? Add the two curves together, and use `lines` again to superimpose the plot of all three curves on the same set of axes. What do you notice about curve that is the sum of the sine and cosine curves?

C1.11 Write an *R* function that computes and plots a sinusoid of user-specified values for the period, amplitude, and phase.

C1.12 One example of a long memory time series is what is called a random walk (See Section 5.1). A simple example of such a process is to let the value of x_t be the number of heads minus the number of tails that have occurred after t flips of a coin. Generate and plot a realization of length 200 from such a process by writing an *R* function that contains the following steps.

(a) Use the `wn` function to generate 200 observations from a uniform(0,1) distribution.

(b) Use the `replace` or `which` function to convert the values of x_t to -1 if $x_t \leq 0.5$ and to $x_t = 1$ if $x_t > 0.5$.

(c) Use the `cum` function to find $S_t = \sum_{j=1}^{t} x_j$ for $t = 1, \ldots, 200$.

(d) Plot S_t versus t.