1.3 R Functions for Basic Statistics

The focus of this *Companion* is on using R for regression analysis, broadly construed. In the course of developing this subject, we will encounter, and indeed already have encountered, a variety of R functions for basic statistical methods (`mean`, `hist`, etc.), but the topic is not addressed systematically.

Table 1.1 shows the names of some standard R functions for basic data analysis. Online help, through `?` or `help`, provides information on the usage of these functions. Where there is a substantial discussion of a function in a later chapter in the present text, the location of the discussion is indicated in the column of the table marked *Reference*. The table is not meant to be complete.

1.4 Generic Functions and Their Methods*

Many of the most commonly used functions in R, such as `summary`, `print`, and `plot`, can have very different actions depending on the arguments passed to the function.[22] For example, the `summary` function applied to different columns of the `Duncan` data frame produces different output. The summary for the variable `Duncan$type` is the count in each level of this factor,

```
> summary(Duncan$type)

  bc prof   wc
  21   18    6
```

while for a numeric variable, the summary includes the mean, minimum, maximum, and several quantiles:

```
> summary(Duncan$prestige)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    3.0    16.0    41.0    47.7    81.0    97.0
```

Similarly, the commands

```
> summary(Duncan)
> summary(lm(prestige ~ income + education, data=Duncan))
```

produce output appropriate to these objects—in the first case by summarizing each column of the `Duncan` data frame and in the second by returning a standard summary for a linear-regression model.

In R, allowing the same *generic function*, such as `summary`, to be used for many purposes is accomplished through an object-oriented programming

[22] The generic `print` function is invoked implicitly and automatically when an object is printed, for example, by typing the name of the object at the R command prompt, or in the event that the object returned by a function isn't assigned to a variable. The `print` function can also be called explicitly, however.

Table 1.1 Some R functions for basic statistical methods. All functions are in the standard R packages; chapter references are to this *Companion.*

Method	*R Function(s)*	*Reference*
histogram	`hist`	Ch. 3
stem-and-leaf display	`stem`	Ch. 3
boxplot	`boxplot`	Ch. 3
scatterplot	`plot`	Ch. 3
time-series plot	`ts.plot`	
mean	`mean`	
median	`median`	
quantiles	`quantile`	
extremes	`range`	
variance	`var`	
standard deviation	`sd`	
covariance matrix	`var`, `cov`	
correlations	`cor`	
normal density, distribution, quantiles, and random numbers	`dnorm`, `pnorm`, `qnorm`, `rnorm`	Ch. 3
t density, distribution, quantiles, and random numbers	`dt`, `pt`, `qt`, `rt`	Ch. 3
chi-square density, distribution, quantiles, and random numbers	`dchisq`, `pchisq`, `qchisq`, `rchisq`	Ch. 3
F density, distribution, quantiles, and random numbers	`df`, `pf`, `qf`, `rf`	Ch. 3
binomial probabilities, distribution, quantiles, and random numbers	`dbinom`, `pbinom`, `qbinom`, `rbinom`	Ch. 3
simple regression	`lm`	Ch. 4
multiple regression	`lm`	Ch. 4
analysis of variance	`aov`, `lm`, `anova`	Ch. 4
contingency tables	`xtabs`, `table`	Ch. 5
generating random samples	`sample`, `rnorm`, etc.	
t-tests for means	`t.test`	
tests for proportions	`prop.test`, `binom.test`	
chi-square test for independence	`chisq.test`	Ch. 5
various nonparametric tests	`friedman.test`, `kruskal.test`, `wilcox.test`, etc.	

technique called *object dispatch*. The details of object dispatch are implemented differently in the S3 and S4 object systems, so named because they originated in Versions 3 and 4, respectively, of the original S language on which R is based.

Almost everything created in R is an object, such as a vector, a matrix, a linear-regression model, and so on.[23] In the S3 object system, which we describe in this section, each object is assigned a *class*, and it is the class of

[23] Indeed, everything in R that is returned by a function is an object, but some functions have *side effects* that create nonobjects, such as files and graphs.

the object that determines how generic functions process the object. We defer consideration of the S4 object system to a later chapter in the book, but it too is class based and implements a version of object dispatch.[24]

The `class` function returns the class of an object:

```
> class(Duncan$type)

[1] "factor"

> class(Duncan$prestige)

[1] "integer"

> class(Duncan)

[1] "data.frame"
```

These objects are of classes `"factor"`, `"integer"`, and `"data.frame"`, consecutively. When the function `lm` is used, an object of class `"lm"` is returned:

```
> duncan.model <- lm(prestige ~ income + education)
> class(duncan.model)

[1] "lm"
```

Generic functions operate on their arguments indirectly by calling specialized functions, referred to as *method functions* or, more compactly, as *methods*. Which method function is invoked typically depends on the class of the first argument to the generic function. For example, the generic `summary` function has the following definition:

```
> summary

function (object, ...)
UseMethod("summary")
<environment: namespace:base>
```

The generic function `summary` has one required argument, `object`, and the special argument `...` (the ellipses) for additional arguments that could be different for each `summary` method. When `UseMethod("summary")` is applied to an object of class `"lm"`, for example, R searches for a method function named `summary.lm` and, if it is found, executes the command `summary.lm(object, ...)`. It is, incidentally, perfectly possible to call `summary.lm` directly; thus, the following two commands are equivalent:

```
> summary(duncan.model)
> summary.lm(duncan.model)
```

Although the generic `summary` function has only one explicit argument, the method function `summary.lm` has additional arguments:

[24] More information on the S3 and S4 object systems is provided in Section 8.7.

```
> args(summary.lm)

function (object, correlation = FALSE, symbolic.cor = FALSE,
    ...)
NULL
```

Because the arguments `correlation` and `symbolic.cor` have default values (`FALSE`, in both cases), they need not be specified. Any additional arguments that are supplied, which are covered by `...`, could be passed to functions that might be called by `summary.lm`.

Although in this instance we can call `summary.lm` directly, many method functions are hidden in the *namespaces* of packages and cannot normally be used directly.[25] In any event, it is good R form to use method functions indirectly through their generics.

Suppose that we invoke the hypothetical generic function `fun` with argument `arg` of class `"cls"`. If there is no method function named `fun.cls`, then R looks for a method named `fun=default`. For example, objects belonging to classes without `summary` methods are printed by `summary.default`. If, under these circumstances, there is no method named `fun.default`, then R reports an error.

We can get a listing of all currently accessible method functions for the generic `summary` using the `methods` function, with hidden methods flagged by asterisks:

```
> methods(summary)

 [1] summary.aov        summary.aovlist    summary.aspell*
 [4] summary.connection summary.data.frame summary.Date
 [7] summary.default    summary.ecdf*      summary.factor
[10] summary.glm        summary.infl       summary.lm
. . .
[25] summary.stl*       summary.table      summary.tukeysmooth*

   Non-visible functions are asterisked
```

These methods may have different arguments beyond the first, and some method functions, for example, `summary.lm`, have their own help pages: `?summary.lm`.

Method selection is slightly more complicated for objects whose `class` is a vector of more than one element. Consider, for example, an object returned by the `glm` function (anticipating a logistic-regression example developed in Section 5.3):

```
> mod.mroz <- glm(lfp ~ ., family=binomial, data=Mroz)
> class(mod.mroz)

[1] "glm" "lm"
```

[25] For example, the `summary` method `summary.loess` is hidden in the namespace of the **stats** package; to call this function directly to summarize an object of class `"loess"`, we could reference the function with the nonintuitive name `stats:::summary=loess`.