1

An Introduction to R

1.1 Overview

R is a rich environment for statistical computing and has many capabilities for exploring data in its base package. In addition, R contains a collection of functions for simulating and summarizing the familiar one-parameter probability distributions. One goal of this chapter is to provide a brief introduction to basic commands for summarizing and graphing data. We illustrate these commands on a dataset about students in an introductory statistics class. A second goal of this chapter is to introduce the use of R as an environment for programming Monte Carlo simulation studies. We describe a simple Monte Carlo study to explore the behavior of the two-sample t statistic when testing from populations that deviate from the usual assumptions. We will find these data analysis and simulation commands very helpful in Bayesian computation.

1.2 Exploring a Student Dataset

1.2.1 Introduction to the Dataset

To illustrate some basic commands for summarizing and graphing data, we consider answers from a sheet of questions given to all students in an introductory statistics class at Bowling Green State University. Some of the questions that were asked included:

1. What is your gender?
2. What is your height in inches?
3. Choose a whole number between 1 and 10.
4. Give the time you went to bed last night.
5. Give the time you woke up this morning.
6. What was the cost (in dollars) of your last haircut, including the tip?
7. Do you prefer water, pop, or milk with your evening meal?
This is a rich dataset that can be used to illustrate methods for exploring a single batch of categorical or quantitative data, for comparing subgroups of the data, such as comparing the haircut costs of men and women, and for exploring relationships.

### 1.2.2 Reading the Data into R

The data for 657 students were recorded in a spreadsheet and saved as the file “studentdata.txt” in text format with tabs between the fields. The first line of the datafile is a header that includes the variable names.

One can read these data into R using the `read.table` command. There are three arguments used in this command. The first argument is the name of the datafile in quotes; the next argument, `sep`, indicates that fields in the file are separated by tab characters; and the `header=TRUE` argument indicates that the file has a header line with the variable names. This dataset is stored in the R data frame called `studentdata`.

```r
> studentdata = read.table("studentdata.txt", sep = ":t", +   header = TRUE)
```

This dataset is also available as part of the `LearnBayes` package. Assuming that the package has been installed and loaded into R, one accesses the data using the `data` command:

```r
> data(studentdata)
```

To see the variable names, we display the first row of the data frame using the `studentdata[1,]` command.

```r
> studentdata[1,]

   Student Height Gender Shoes Number Dvds ToSleep WakeUp Haircut Job Drink
    1     67 female  10     5  10   -2.5   5.5    60   30   water
```

To make the variable names visible in the R environment, we use the `attach` command.

```r
> attach(studentdata)
```

### 1.2.3 R Commands to Summarize and Graph a Single Batch

One categorical variable in this dataset is `Drink`, which indicates the student’s drinking preference between milk, pop, and water. One can tally the different responses using the `table` command.

```r
> table(Drink)
```
Drink
milk  pop  water
113   178  355

We see that more than half the students preferred water, and pop was more popular than milk.

One can graph these frequencies with a bar graph using the `barplot` command. We first save the output of `table` in the variable `t` and then use `barplot` with `t` as an argument. We add labels to the horizontal and vertical axes by the `xlab` and `ylab` argument options. Figure 1.1 displays the resulting graph.

```r
> table(Drink)
> barplot(table(Drink),xlab="Drink",ylab="Count")
```

![Barplot](image)

**Fig. 1.1.** Barplot of the drinking preference of the statistics students.

Suppose we are next interested in examining how long the students slept the previous night. We did not directly ask the students about their sleeping time, but we can compute a student’s hours of sleep by subtracting her go-to-bed time from her wake-up time. In R we perform this computation for all students, and the variable `hours.of.sleep` contains the sleeping times.
> hours.of.sleep = WakeUp - ToSleep

A simple way to summarize this quantitative variable uses the `summary` command, which gives a variety of descriptive statistics about the variable.

> summary(hours.of.sleep)

```
   Min. 1st Qu.  Median     Mean  3rd Qu.     Max.     NA's
   2.500  6.500  7.500    7.385  8.500  12.500    4.000
```

On average, we see that students slept 7.5 hours and half of the students slept between 6.5 and 8.5 hours.

To see the distribution of sleeping times, we can construct a histogram using the `hist` command (see Figure 1.2).

> hist(hours.of.sleep,main="")

The shape of this distribution looks symmetric about the average value of 7.5 hours.

![Histogram of the hours of sleep of the statistics students.](image)

**Fig. 1.2.** Histogram of the hours of sleep of the statistics students.
1.2.4 R Commands to Compare Batches

Since the gender of each student was recorded, one can make comparisons between men and women on any of the quantitative variables. Do men tend to sleep longer than women? We can answer this question graphically by constructing parallel boxplots of the sleeping times of men and women. Parallel boxplots can be displayed using the `boxplot` command. The argument is given by

```
hours.of.sleep ~ Gender
```

This indicates that a boxplot of the hours of sleep will be constructed for each level of `Gender`. The resulting graph is displayed in Figure 1.3. From the display, it appears that men and women are similar with respect to their sleeping times.

```r
> boxplot(hours.of.sleep~Gender,
+  ylab="Hours of Sleep")
```

![Parallel boxplots of the hours of sleep of the male and female students.](image)

**Fig. 1.3.** Parallel boxplots of the hours of sleep of the male and female students.

For other variables, there are substantial differences between the two genders. Suppose we wish to divide the haircut prices into two groups – the
haircut prices for the men and the haircut prices for the women. We do this using the R logical operator `==`. The syntax

```r
Gender=="female"
```

is a logical statement that will be `TRUE` if `Gender` is “female”; otherwise it will be `FALSE`. The expression

```r
Haircut[condition]
```

will produce a subset of `Haircut` according to when the `condition` is `TRUE`. So the statement

```r
> female.Haircut=Haircut[Gender=="female"]
```

will select the haircut prices only for the female students and store the prices into the variable `female.Haircut`. Similarly, we use the logical operator to store the male haircut prices into the variable `male.Haircut`.

```r
> male.Haircut=Haircut[Gender=="male"]
```

By using the `summary` command, we summarize the haircut prices of the women and the men.

```r
> summary(female.Haircut)
```

```
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
0.00 15.00 25.00 34.08 45.00 180.00 19.00
```

```r
> summary(male.Haircut)
```

```
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
0.00 0.00 12.00 10.54 15.00 75.00 1.00
```

We see large differences between men and women – the men average about $10 for a haircut and the women average about $34.

### 1.2.5 R Commands for Studying Relationships

There are many interesting relationships that can be explored in this student dataset. To get a good night’s sleep, one may want to go to bed early in the evening. This raises the question: “Is the length of sleep for a student related to the time that he or she goes to bed?” We can explore the relationship between the `ToSleep` and `hours.of.sleep` variables by means of a scatterplot. The R command `plot(ToSleep,hours.of.sleep)` will construct a scatterplot with `ToSleep` on the horizontal scale and `hours.of.sleep` on the vertical scale. If we draw this scatterplot, it is a little difficult to see the pattern in the graph since many of the points are identical. We use the `jitter` function on each variable before plotting – this has the effect of adding a small amount of noise so that more points are visible on the graph (see Figure 1.4).
1.2 Exploring a Student Dataset

We can describe the decreasing pattern in this scatterplot by fitting a line. A least-squares fit is done using the \texttt{lm} command; this has the syntax

\begin{verbatim}
> fit=lm(hours.of.sleep~ToSleep)
\end{verbatim}

The output of this fitting is stored in the variable \texttt{fit}. If we display this variable, we see the intercept and slope of the least-squares line.

\begin{verbatim}
> fit
Call:
   lm(formula = hours.of.sleep ~ ToSleep)
Coefficients:
    (Intercept)       ToSleep
       7.9628       -0.5753
\end{verbatim}

The slope is approximately \(-0.5\), which means that a student loses about a half hour of sleep for every hour later that he or she goes to bed.

We can display this line on top of the scatterplot by using the \texttt{abline} command (see Figure 1.5), where the single argument is the variable \texttt{fit}.
> abline(fit)

![Scatterplot of wake-up time and hours of sleep for students with least-squares line plotted on top.](image)

**Fig. 1.5.** Scatterplot of wake-up time and hours of sleep for students with least-squares line plotted on top.

### 1.3 Exploring the Robustness of the *t* Statistic

#### 1.3.1 Introduction

Suppose one has two independent samples, $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$, and wishes to test the hypothesis that the mean of the $x$ population is equal to the mean of the $y$ population:

$$ H_0 : \mu_x = \mu_y. $$

Let $\bar{X}$ and $\bar{Y}$ denote the sample means of the $x$s and $y$s and let $s_x$ and $s_y$ denote the respective standard deviations. The standard test of this hypothesis $H_0$ is based on the *t* statistic

$$ T = \frac{\bar{X} - \bar{Y}}{s_p \sqrt{1/m + 1/n}}, $$
where \( s_p \) is the pooled standard deviation

\[
s_p = \sqrt{\frac{(m-1)s_x^2 + (n-1)s_y^2}{m+n-2}}.
\]

Under the hypothesis \( H_0 \), the test statistic \( T \) has a \( t \) distribution with \( m+n-2 \) degrees of freedom when

- both the \( x \)s and \( y \)s are independent random samples from normal distributions
- the standard deviations of the \( x \) and \( y \) populations, \( \sigma_x \) and \( \sigma_y \), are equal

Suppose the level of significance of the test is set at \( \alpha \). Then one will reject \( H \) when

\[
|T| \geq t_{n+m-2,\alpha/2},
\]

where \( t_{df,\alpha} \) is the \((1 - \alpha)\) quantile of a \( t \) random variable with \( df \) degrees of freedom.

If the underlying assumptions of normal populations and equal variances hold, then the level of significance of the \( t \)-test will be the stated level of \( \alpha \). But, in practice, many people use the \( t \) statistic to compare two samples even when the underlying assumptions are in doubt. So an interesting problem is to investigate the robustness or sensitivity of this popular test statistic with respect to changes in the assumptions. If the stated significance level is \( \alpha = .10 \) and the populations are skewed or have heavy tails, what will be the true significance level? If the assumption of equal variances is violated and there are significant differences in the spreads of the two populations, what is the true significance level? One can answer these questions using a Monte Carlo simulation study. R is a very suitable platform for writing a simulation algorithm. One can generate random samples from a wide variety of probability distributions, and R has an extensive set of data analysis capabilities for summarizing and graphing the simulation output. Here we illustrate the construction of a simple R function to address the robustness of the \( t \) statistic.

### 1.3.2 Writing a Function to Compute the \( t \) Statistic

To begin, we generate some random data for the samples of \( x \)s and \( y \)s. We simulate a sample of ten observations from a normal distribution with mean 50 and standard deviation 10 using the `rnorm` function and store the vector of values in the variable \( x \). Likewise we simulate a sample of \( y \)s by simulating ten values from an \( N(50, 10) \) distribution and store these values in the variable \( y \).

\[
> x=rnorm(10,\text{mean}=50,\text{sd}=10) \\
> y=rnorm(10,\text{mean}=50,\text{sd}=10)
\]

Next we write a few lines of R code to compute the value of the \( t \) statistic from the samples in \( x \) and \( y \). We find the sample sizes \( m \) and \( n \) by using the R command `length`.
We compute the pooled standard deviation $s_p$ – in the R code, `sd` is the standard deviation function and `sqrt` takes the square root of its argument.

\[
s_p = \sqrt{\frac{(m-1) \cdot \text{sd}(x)^2 + (n-1) \cdot \text{sd}(y)^2}{m+n-2}}
\]

With `m`, `n`, and `sp` defined, we compute the t statistic

\[
t.stat = \frac{\text{mean}(x) - \text{mean}(y)}{sp \cdot \sqrt{\frac{1}{m} + \frac{1}{n}}}
\]

By combining these R statements, we can write a short R function `tstatistic` to compute the t statistic. This function has two arguments, the vectors `x` and `y`, and the output of the function (indicated by the `return` statement) is the value of the t statistic.

```r
tstatistic=function(x,y)
{
  m=length(x)
  n=length(y)
  sp=sqrt(((m-1)*sd(x)^2+(n-1)*sd(y)^2)/(m+n-2))
  t.stat=(mean(x)-mean(y))/(sp*sqrt(1/m+1/n))
  return(t.stat)
}
```

Suppose this function has been saved in the file “tstatistic.R”. We enter this function into R by using the `source` command.

```r
> source("tstatistic.R")
```

We try the function by placing some fake data in vectors `data.x` and `data.y` and then computing the t statistic on these data:

```r
> data.x=c(1,4,3,6,5)
> data.y=c(5,4,7,6,10)
> tstatistic(data.x, data.y)
[1] -1.937926
```

### 1.3.3 Programming a Monte Carlo Simulation

Suppose we are interested in learning about the true significance level for the t statistic when the populations don’t follow the standard assumptions of normality and equal variances. In general, the true significance level will depend on

- the stated level of significance $\alpha$
- the shape of the populations (normal, skewed, heavy-tailed, etc.)
- the spreads of the two populations as measured by the two standard deviations
Given a particular choice of $\alpha$, shape, spreads, and sample sizes, we wish to estimate the true significance level given by

$$\alpha^T = P(|T| \geq t_{n+m-2,\alpha/2}).$$

Here is an outline of a simulation algorithm to compute $\alpha^T$:

1. Simulate a random sample $x_1, \ldots, x_m$ from the first population and $y_1, \ldots, y_n$ from the second population.
2. Compute the t statistic $T$ from the two samples.
3. Decide if $|T|$ exceeds the critical point and $H_0$ is rejected.

One repeats steps 1–3 of the algorithm $N$ times. One estimates the true significance level by

$$\hat{\alpha}^T = \frac{\text{number of rejections of } H_0}{N}.$$

The following is an R script that implements the simulation algorithm for normal populations with mean 0 and standard deviation 1. The R variable `alpha` is the stated significance level, $m$ and $n$ are the sample sizes, and $N$ is the number of simulations. The `rnorm` command is used to simulate the two samples and $T$ contains the value of the t statistic. One decides to reject if

$$\text{abs}(t) > \text{qt}(1-\alpha/2,n+m-2)$$

where `qt(p, df)` is the $p$th quantile of a t distribution with $df$ degrees of freedom. The observed significance level is stored in the variable `true.sig.level`.

```r
alpha=.1; m=10; n=10  # sets alpha, m, n
N=10000  # sets the number of simulations
n.reject=0  # counter of num. of rejections
for (i in 1:N)
{
x=rnorm(m,mean=0,sd=1)  # simulates xs from population 1
y=rnorm(n,mean=0,sd=1)  # simulates ys from population 2
t.stat=tstatistic(x,y)  # computes the t statistic
if (abs(t.stat)>qt(1-alpha/2,n+m-2))
    n.reject=n.reject+1  # reject if |T| exceeds critical pt
}
true.sig.level=n.reject/N  # est. is proportion of rejections
```

### 1.3.4 The Behavior of the True Significance Level Under Different Assumptions

The R script described in the previous section can be used to explore the pattern of the true significance level $\alpha^T$ for different choices of sample sizes and populations. The only two lines that need to be changed in the R script...
are the definition of the sample sizes $m$ and $n$ and the two lines where the two samples are simulated.

Suppose we fix the stated significance level at $\alpha = .10$ and keep the sample sizes at $m = 10$ and $n = 10$. We simulate samples from the following populations, where the only restriction is that the population means be equal:

- Normal populations with zero means and equal spreads ($\sigma_x = \sigma_y = 1$)
  
  $x = \text{rnorm}(m, \text{mean}=0, \text{sd}=1)$
  $y = \text{rnorm}(n, \text{mean}=0, \text{sd}=1)$

- Normal populations with zero means and very different spreads ($\sigma_x = 1, \sigma_y = 10$)
  
  $x = \text{rnorm}(m, \text{mean}=0, \text{sd}=1)$
  $y = \text{rnorm}(n, \text{mean}=0, \text{sd}=10)$

- T populations, 4 degrees of freedom, and equal spreads
  
  $x = \text{rt}(m, \text{df}=4)$
  $y = \text{rt}(n, \text{df}=4)$

- Exponential populations with $\mu_x = \mu_y = 1$
  
  $x = \text{rexp}(m, \text{rate}=1)$
  $y = \text{rexp}(n, \text{rate}=1)$

- One normal population ($\mu_x = 10, \sigma_x = 2$) and one exponential population ($\mu_y = 10$).
  
  $x = \text{rnorm}(m, \text{mean}=10, \text{sd}=2)$
  $y = \text{rexp}(n, \text{rate}=1/10)$

The R script was run for each of these five population scenarios using $N = 10000$ iterations, and the estimated true significance levels are displayed in Table 1.1. These values should be compared with the stated significance level of $\alpha = .1$, keeping in mind that the simulation standard error of each estimate is equal to $\sqrt{\frac{1(0.9)}{10000}} = 0.003$. (The simulation standard error, the usual standard error for a binomial proportion, is equal to $\sqrt{\frac{1(0.9)}{10000}} = 0.003$.) In this brief study, it appears that if the populations have equal spreads, then the true significance level is approximately equal to the stated level for different population shapes. If the populations have similar shapes and different spreads, then the true significance level can be slightly higher than 10%. If the populations have substantially different shapes (such as normal and exponential) and unequal spreads, then the true significance level can be substantially higher than the stated level.

Since the true significance level in the last case is 50% higher than the stated level, one might be interested in seeing the exact sampling distribution of the t statistic. We rerun this simulation for the normal and exponential populations. First we define the sample sizes $m$ and $n$ and write a short function my.tsimulation that computes the t statistic for the simulated data.
Table 1.1. True significance levels of the t-test computed by Monte Carlo experiments. The standard error of each estimate is approximately 0.003.

<table>
<thead>
<tr>
<th>Populations</th>
<th>True Significance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal populations with equal spreads</td>
<td>0.0986</td>
</tr>
<tr>
<td>Normal populations with unequal spreads</td>
<td>0.1127</td>
</tr>
<tr>
<td>t(4) distributions with equal spreads</td>
<td>0.0968</td>
</tr>
<tr>
<td>Exponential populations with equal spreads</td>
<td>0.1019</td>
</tr>
<tr>
<td>Normal and exponential populations with unequal spreads</td>
<td>0.1563</td>
</tr>
</tbody>
</table>

> m=10; n=10
> my.tsimulation=function()
+ tstatistic(rnorm(m,mean=10,sd=2), rexp(n,rate=1/10))

Then we repeat this simulation 10,000 times using the `replicate` function.

> tstat.vector=replicate(10000, my.tsimulation())

The simulated values of the t statistic are stored in the vector `tstat.vector`.

We use the R command `density` to construct a nonparametric density estimate of the exact sampling distribution of the t statistic. The `curve` command is used to plot the t density with 18 degrees of freedom on top. Figure 1.6 displays the resulting graph of the two densities. Note that the actual sampling distribution of the t statistic is right-skewed, which would account for the large true significance level.

> plot(density(tstat.vector), xlim=c(-5,8), ylim=c(0,.4), lwd=3)
> curve(dt(x,df=18), add=TRUE)
> legend(4,.3,c("exact","t(18)"), lwd=c(3,1))

1.4 Further Reading

Although R is a sophisticated package with many commands, there are many resources available for learning the package. Some basic instruction on R can be found from the R Help menu. The R project home page at http://www.r-project.org lists a number of books describing different levels of statistical computing using R. Verzani (2004) is a good book describing the use of R in an introductory statistics course; in particular, the book is helpful for getting started in constructing different types of graphical displays. Appendix A in Gentle (2002) gives a general description of Monte Carlo experiments with an extended example.
Fig. 1.6. Exact sampling density of the t statistic when sampling from normal and exponential distributions. The t sampling density assuming normal populations is also displayed.

1.5 Summary of R Functions

An outline of the R functions used in this chapter is presented here. Detailed information about any specific function, say `abline`, can be found by typing `?abline` in the R command window.

`abline` – add a straight line to a plot

`attach` – attach a set of R objects to the search path

`barplot` – create a barplot with vertical or horizontal bars

`boxplot` – produce box-and-whisker plot(s) of the given (grouped) values

`density` – computes kernel density estimates

`hist` – computes a histogram of the given data values

`lm` – used to fit linear models such as regression
mean – computes the arithmetic mean

plot – generic function for plotting R objects

read.table – reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file

rexp – random generation for the exponential distribution

rnorm – random generation for the normal distribution

rt – random generation for the t distribution

sd – computes the value of the standard deviation

summary – generic function used to produce result summaries of the results of various model-fitting functions

table – uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels

1.6 Exercises

1. Movie DVDs owned by students
   The variable Dvds in the student dataset contains the number of movie DVDs owned by students in the class.
   a) Construct a histogram of this variable using the hist command.
   b) Summarize this variable using the summary command.
   c) Use the table command to construct a frequency table of the individual values of Dvds that were observed. If one constructs a barplot of these tabled values using the command
      \[
      \text{barplot(table(Dvds))}
      \]
      one will see that particular response values are very popular. Is there any explanation for these popular values for the number of DVDs owned?

2. Student heights
   The variable Height contains the height (in inches) of each student in the class.
   a) Construct parallel boxplots of the heights using the Gender variable.
   b) If one assigns the boxplot output to a variable
      \[
      \text{output=boxplot(Height~Gender)}
      \]
      then output is a list that contains statistics used in constructing the boxplots. Print output to see the statistics that are stored.
   c) On average, how much taller are male students than female students?

3. Sleeping times
   The variables ToSleep and WakeUp contain, respectively, the time to bed and wake-up time for each student the previous evening. (The data are recorded as hours past midnight, so a value of −2 indicates 10 p.m.)
a) Construct a scatterplot of ToSleep and WakeUp.
b) Find a least-squares fit to these data using the `lm` command.
c) Place the least-squares fit on the scatterplot using the `abline` command.
d) Use the line to predict the wake-up time for a student who went to bed at midnight.

### 4. Performance of the traditional confidence interval for a proportion
Suppose one observes $y$ that is binomially distributed with sample size $n$ and probability of success $p$. The standard 90% confidence interval for $p$ is given by

$$ C(y) = \left( \hat{p} - 1.645 \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \hat{p} + 1.645 \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \right), $$

where $\hat{p} = y/n$. We use this procedure under the assumption that $P(p \in C(y)) = 0.90$ for all $0 < p < 1$.

The function `binomial.conf.interval` will return the limits of a 90% confidence interval given values of $y$ and $n$.

```r
binomial.conf.interval=function(y,n)
{
  z=qnorm(.95)
  phat=y/n
  se=sqrt(phat*(1-phat)/n)
  return(c(phat-z*se,phat+z*se))
}
```

a) Read the function `binomial.conf.interval` into R.
b) Suppose that samples of size $n = 20$ are taken and the true value of the proportion is $p = .5$. Using the `rbinom` command, simulate a value of $y$ and use `binomial.conf.interval` to compute the 90% confidence interval. Repeat this a total of 20 times, and estimate the true probability of coverage $P(p \in C(y))$.

c) Suppose that $n = 20$ and the true value of the proportion is $p = .05$. Simulate 20 binomial random variates with $n = 20$ and $p = .05$, and for each simulated $y$ compute a 90% confidence interval. Estimate the true probability of coverage.

### 5. Performance of the traditional confidence interval for a proportion
Exercise 4 demonstrated that the actual probability of coverage of the traditional confidence interval depends on the values of $n$ and $p$. Construct a Monte Carlo study that investigates how the probability of coverage depends on the sample size and true proportion value. In the study, let $n$
be 10, 25, and 100 and let \( p \) be .05, .25, and .50. Write an R function that has three inputs, \( n \), \( p \), and the number of Monte Carlo simulations \( m \), and will output the estimate of the exact coverage probability. Implement your function using each combination of \( n \) and \( p \) and \( m = 1000 \) simulations. Describe how the actual probability of coverage of the traditional interval depends on the sample size and true proportion value.